LaTenS: Final Report

Daniel Schwartz	**Eliana Ward-Lev**	**Mohit Rajpal**	**Elsbeth Turcan**
Manager	*Language Guru*	*System Architect*	*Tester*
ds3263	erw2138	mr3522	ect2150

Published: December 21, 2016

Contents

1 Introduction

1.1 Introduction to Tensors

"Pure mathematics is, in its way, the poetry of logical ideas." - Albert Einstein

Tensors are essentially the generalization of a matrix or vector to arbitrary rank. Rank refers to the number of indices needed to label a component of that tensor. For example, a scalar is a rank zero tensor, a vector is a rank one tensor and a matrix is a rank two tensor.
Indices are used to denote a particular element of a tensor. For example, the formula to find the eigenvectors of a matrix M is

$$\begin{pmatrix} M_{00} & M_{01} & M_{02} \\ M_{10} & M_{11} & M_{12} \\ M_{20} & M_{21} & M_{22} \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ v_2 \end{pmatrix} = \lambda \begin{pmatrix} v_0 \\ v_1 \\ v_2 \end{pmatrix}$$

Using index notation, the eigenvector formula becomes

$$\sum_{j=0}^{3} M_{ij} v_j = \lambda v_i$$

Einstein summation convention allows for dropping of the sign

$$M_{ij} v_j = \lambda v_i$$

Note that Einstein summation follows a simple rule: any pair of indices with the same symbol implies summation. Also note that any index without a pair must also appear on the other side of the equality (here i).
Tensors are defined under addition, subtraction and multiplication. Addition and subtraction are both performed element-wise. Multiplication uses the implied summation discussed above.
In more detail, to multiply two (2X2) tensors $T1_{ab}T2_{bc}$(this is standard matrix multiplication), one gets a (2X2) matrix

$$\begin{pmatrix} \sum_b T1_{0b}T2_{b0} & \sum_b T1_{0b}T2_{b1} \\ \sum_b T1_{1b}T2_{b0} & \sum_b T1_{1b}T2_{b1} \end{pmatrix}$$

Here both tensors are rank-two and only one index is bound(b), but this generalizes to arbitrary rank and an arbitrary number of bound indices(with a sum over each of the bound indices to find each element). In order to multiply a rank-n tensor by a rank-m matrix each dimension which is bound must have the same size as the dimension in the other matrix which it is bound to. Multiplying a rank-n matrix by a rank-m matrix with b bound indices results in a rank-$(n + m - 2b)$ tensor.
One can also access lower rank slices of tensors by specifying one or more indices and leaving some(or none) free. If all indices are specified, it is a rank-0 tensor which is the same thing as a scalar.

1.2 Motivation

"Tensors are mathematical objects that can be used to represent real-world systems. [...] Tensors have proven to be useful in many engineering contexts, in fluid dynamics, and for the General Theory of Relativity. Knowledge of tensor math [...] also is useful in the fields of financial analysis, machine [learning], and in the analysis of other complex systems." [1]

Many fields of physics rely on tensor math. Notably, tensors are central in Einstein's theory of general relativity. The most important equation in General Relativity is a tensor equation relating a set of symmetric 4×4 tensors.

$$R_{\mu\nu} - \frac{1}{2} R g_{\mu\nu} + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu}$$

where $R_{\mu\nu}$ describes the curvature of space-time, $g_{\mu\nu}$ is the metric tensor which is what one needs to solve for, λ is the cosmological constant, G is Newton's gravitational constant, c is the speed of light in vacuum, and $T_{\mu\nu}$ describes the matter present. Tensors are needed both to solve this equation and to use the results to solve problems like figuring out how time dilation effects GPS satellites.

Tensors are also important constructs in machine learning and related fields. Nearly every construct used in any machine learning application is a tensor–for example, it is most common to represent data points as n-dimensional vectors, and the same is true for linear classifiers. Higher-dimensional tensors have found uses across algorithms and applications as well, as they often provide the most intuituve way to express some data (imagine a series of 2-D images in a video, for example), and linear algebra is quick to execute as modern processor speeds and memory capacities improve. For evidence of their ubiquity, simply recall that Google named one of today's most popular machine learning backends "TensorFlow".

1.3 Goals

LaTenS is a programming language designed to be easy to use for computer scientists as well as mathematicians and physicists. The syntax used resembles that of LaTeX, and C both of which should be familiar to these users. Additionally, we provide a second compiler which by using a flag compiles the code into a document which can be turned into a PDF using one's favorite LaTeXcompiler. This makes it easier to people to share their code in a readable document.

2 Language Tutorial

2.1 Language Description

LaTenS is a language that strives to make calculations with tensors as simple as possible. We generalize a matrix-manipulation language to manipulate n-dimensional tensors. The syntax is intended to evoke traditional mathematical syntax for operating on tensors—for example, when composing in LaTenS, an element of a tensor T_{00} can be accessed by typing `T_{0,0}`, so this syntax is likely familiar to mathematicians.

LaTenS uses this kind of syntax to make problems involving tensor manipulation quick and intuitive, with built-in operations on n-dimensional tensors without the need to loop over elements in the tensors (as in other programming languages).

2.2 Using the Compiler

Inside the directory 'LaTenS' type make. This creates the latens compiler that takes in '.tens' files and compiles them to corresponding '.ll' files corresponding to LLVM IR. The syntax for running the dice executable is: ./latens.native [optional-option] <sourcefile.tens> — lli-3.8. There are also additional flags with respect to the compiler that allow for additional options.

- "-a" - Print the AST only
- "-p" - Pretty print only (compile code to LaTeX)
- "-s" - Semantically check and print SAST (semantically-checked AST)
- "-d" - Pretty print the SAST and also the final global environment
- "-l" - Generate LLVM, don't check
- "-c" - Generate, check LLVM IR

2.3 Hello World Example

LaTenS programs are composed of the following elements

- functions
- variables
- statements
- expressions
- comments

Lets look at simple example of the code that would print "Hello world!"

```
int main () {
    %simple LaTenS program
    print("Hello world!\n");
    return 0;
}
```

Looking at this line by line:

- The first line `int main (){` is the main function where the code begins. All programs must have a main function.
- The second line % ... is a comment and will be ignored by the compiler
- The third line uses a built in function 'print' to display the message "Hello world!" to the screen
- The final line returns 0 and ends the main function.

2.4 Tokens

Programs in LaTenS consist of tokens. A token can be a keyword, an identifier, a number, string literal or a symbol.

2.4.1 Comments

Anything after a % on the same line or between %{....%} is a comment. Comments are ignored by the compiler and are simply used to help readers of your code. You cannot have comments within comments or within string literals.

2.4.2 Identifiers

Identifiers are used to reference variables and functions. A valid identifier is a letter (uppercase or lowercase) followed by any number of uppercase or lowercase letters and digits. No other characters may be included in an identifier.

2.4.3 Whitespace

Whitespace (spaces tabs and newlines) is ignored by the LaTenS compiler. Use whitespace to benefit the readability of your code.

2.4.4 Semicolons

Each non-block statement must be terminated by a semicolon.

2.4.5 Types

There are three basic types in LaTenS: int, float and string. There is also a void type and tensor types. Integers and floats are both signed 64 bit numbers. Void can only be used to declare functions with no return. Tensors store rank in their type, but can be of arbitrary rank.

3 Variables

3.1 Declaration

A variable declaration sets an identifier equal to an expression.

```
let a = 5;
let b = 4.5;
let c = "hello";
let T1 = [1, 2, 3];
let T2_{2, 2};
```

Each variable declaration must contain the keyword 'let' and an identifier followed by '=' and an expression. Tensors can be declared with or without indices. With indices, they can be declared without being assigned (in every other instance an assingment must be provided).

3.2 Example

```
%% A pithy code sample that shows us
%% how much cattle we would have if we
%% bought 40 cattle some number of years ago.
%% With the help of our friends: https://www.wku.edu/mathmatters/documents/mathmattersep13.pdf
%% (ie we implemented their stuff in LaTenS)
int main () {
    print("Suppose you purchase 40 head of 2 year old cattle\n");

    %% Our Leslie Matrix, for heads of cattle
    let L = [[ 0, .5, 1.1, 1.3, 0.8, .4],
             [.7, 0, 0, 0, 0, 0],
             [0, .9, 0, 0, 0, 0],
             [0, 0, .8, 0, 0, 0],
             [0, 0, 0, .7, 0, 0],
             [0, 0, 0, 0, .3, 0]];

    %% We've got 40 cattle at t=0; where t is measured in years
    let D0 = [0, 40, 0, 0, 0, 0];


    print("given survival rates and birth rates: \n");
    print(L);


    print("you want to know what your age distribution is\n");
    print("after 2 years: ");

    %% Each multiplication moves t forward 2 years.
    %% So now t=2
    let D = L_{a, b}*D0_{b};
```

```
    print(D);


    %% Let's jump t forward 28 more years
    %% So t = 2 + 28 = 30
    for(let i = 0; i < 14; i = i+1){
        D = L_{a, b}*D_{b};
    }
    print("after 30 years: ");
    print(D);

     %% Let's jump t forward 28 more years
    %% So t = 2 + 28 = 30
    for(let i = 0; i < 14; i = i+1){
        D = L_{a, b}*D_{b};
    }
    print("after 30 years: ");
    print(D);


    %% And of course, no demo would be complete
    %% without showing t=42
    for(let i = 0; i < 26; i = i+1){
        D = L_{a, b}*D_{b};
    }
    print("after 42 years: ");
    print(D);

    return 0;
}
```

Note: comments with %% are included in the code compiled to LaTeX.
This code consisists of a main method. It declares a 2-tensor L and a two 1-tensors D0 and D. It uses three for loops(which are set to run first 14 times, then 14 times again then 26 times. It outputs

```
Suppose you purchase 40 head of 2 year old cattle
given survival rates and birth rates:
0.000000, 0.500000, 1.100000, 1.300000, 0.800000, 0.400000,
0.700000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 0.900000, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000, 0.800000, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000, 0.000000, 0.700000, 0.000000, 0.000000,
0.000000, 0.000000, 0.000000, 0.000000, 0.300000, 0.000000,

you want to know what your age distribution is
after 2 years: 20.000000, 0.000000, 36.000000, 0.000000, 0.000000, 0.000000,
after 30 years: 482.239451, 274.548626, 200.689070, 130.048246, 74.073542, 18.184948,
after 42 years: 107014.740228, 60856.805449, 44495.815173, 28918.559794, 16445.315656, 4008.030174,
```

3.3 LaTeXPrinting Example

```
%% An example of nice looking code
%% using our built in \LaTeX pretty printer

```

```
int main () {
    let bowtie = [[[[[[[[0]]]]]]]];
    let T = [[1, 0, 0, 0, 0, 0],
             [0, 1, 0, 0, 0, 0],
             [0, 0, 1, 0, 0, 0],
             [0, 0, 0, 1, 0, 0],
             [0, 0, 0, 0, 1, 0],
             [0, 0, 0, 0, 0, 1]];
    let a = (5*3+7)/(4-156);
    return 0;
}
```

Becomes:

An example of nice looking code
using our built in LaTeXpretty printer

```
int main()
{
    let bowtie = [[[[[[[[0]]]]]]]];
    let T = [[1, 0, 0, 0, 0, 0],
             [0, 1, 0, 0, 0, 0],
             [0, 0, 1, 0, 0, 0],
             [0, 0, 0, 1, 0, 0],
             [0, 0, 0, 0, 1, 0],
             [0, 0, 0, 0, 0, 1]];
    let a = (5*3+7)/(4-156);

    return 0;
}
```

4 Language Reference Manual

4.1 Lexical Conventions

4.1.1 Tokens

A LaTenS program is parsed into tokens according to the following scanner. Most important are the operators, and then the reserved keywords. Comments and whitespace are ignored (except to separate tokens and special comments designed to appear in LaTeX-formatted pretty-printouts).

```
let alpha = ['a'-'z' 'A'-'Z']
let ascii = ([' '-'!' '#'-'[' ']'-'~'])
let digit = ['0'-'9']
let id = alpha (alpha | digit)*
let float = digit* ('.' digit*)? (['e' 'E'] ['-' '+']? digit+)?
let int = digit+
```

```
let whitespace = [' ' '\t' '\r']
let escape = '\\' ['\\' '\'' '"' 'n' 'r' 't']
let string = '"' ((escape | [^ '"' '\\'])* as str) '"'



rule token = parse
  [' ' '\t' '\r'] { token lexbuf } (* Whitespace *)
| '\n'     { Lexing.new_line lexbuf; token lexbuf } (* Newline *)
| "%{"     { comment lexbuf } (* Block comments *)
| "%"    { singlecomment lexbuf } (* Single-line comments *)
| ("%%" [^ '\n' ]*) as lxm { COMMENT(string_of_int lexbuf.lex_curr_p.pos_lnum, match String.length
    lxm with 0 | 1 | 2 -> "" | x -> String.sub lxm 2 (x-2)) }
| '('    { LPAREN }
| ')'    { RPAREN }
| '['    { LBRACKET }
| ']'      { RBRACKET }
| '{'    { LBRACE }
| '}'    { RBRACE }
| ';'    { SEMI }
| ','    { COMMA }
| '_'    { UNDER }
| '+'    { PLUS }
| '-'    { MINUS }
| '*'    { TIMES }
| '/'    { DIVIDE }
| '^'    { CONCAT }
| "=="     { EQ }
| '='    { ASSIGN }
| "~="     { NEQ }
| '<'    { LT }
| "<="     { LEQ }
| '>'    { GT }
| ">="     { GEQ }
| "&&"     { AND }
| "||"     { OR }
| '~'    { NOT }
| "else" { ELSE }
| "int"    { INT }
| "float"  { FLOAT }
| "void" { VOID }
| "for"    { FOR }
| "if"     { IF }
| "let"    { LET }
| "return" { RETURN }
| "string" { STRING }
| "while"  { WHILE }
| int as lxm { ILITERAL(string_of_int (lineno lexbuf), int_of_string lxm) }
| float as lxm { FLITERAL(string_of_int (lineno lexbuf), float_of_string lxm) }
| string {
   String.iter (fun n-> match n with '\n' -> Lexing.new_line lexbuf | _ -> ()) str;
   SLITERAL(string_of_int (lineno lexbuf), unescape str) }
| id as lxm { ID(string_of_int (lineno lexbuf), lxm) }
| eof { EOF }
| '"'      { raise (Exceptions.UnmatchedQuotationException(lineno lexbuf)) }
| _ as illegal { raise (Exceptions.IllegalCharacterException(lineno lexbuf, illegal)) }
```

```
and comment = parse
  "%}" { token lexbuf }
| '\n' { Lexing.new_line lexbuf; comment lexbuf } (* Increment line counter *)
| _ { comment lexbuf }

and singlecomment = parse
  ('\n') { Lexing.new_line lexbuf; token lexbuf } (* Single-line comment ends with a newline or a
      carriage return*)
| "%{" { comment lexbuf } (* If we see a multiline comment,
                          forget about looking for a newline *)
| _ { singlecomment lexbuf }
```

4.1.2 Comments

LaTenS supports single line comments

```
% Single Line Comment
```

and block comments over multiple lines

```
%{ Block Comment %}
```

Comments may not be nested, although the two types may be combined.

```
%{
    % This comment structure is also valid,
    % Although really everything inside a block comment is ignored,
    % So the single-line comments are not actually being recognized.
    %% And this comment will not make it into a oretty-printed LaTeX document.
%}
```

4.1.3 Identifiers

An identifier name must be a letter (uppercase or lowercase) followed by any number of uppercase or lowercase letters and digits. No other characters may be included in an identifier; note that in particular, programmers may expect the underscore (_) to be a legal identifier character, but see Section 5.2.2 (Tensor) for an explanation of why the underscore character is reserved.

4.1.4 Key Words

No identifier name can be a keyword reserved by the system. The following words are used elsewhere (control flow, for example) and are reserved:

- else
- float
- int
- void
- for
- if

- let
- return
- string
- while

Reserved words are case-sensitive, so for example "If" is a valid identifier name.

4.2 Data Types

```
typ:
    FLOAT { Float }
  | STRING {String}
  | VOID { Void }
  | INT { Int }
  | ID UNDER LBRACE qual_list_expr { TensorType($4) }
```

```
var_typ:
   FLOAT { Float }
 | STRING { String }
 | INT { Int }
 | ID UNDER LBRACE qual_list_expr { TensorType($4) }
```

LaTenS is statically typed with hybrid type-checking, following the conventions of Swift or GO. This implies that variable types are inferred, but function return values and formal argument types must be specified.

Every variable has a type (e.g. rank n tensor or string) and can be used in assignment as long as the right-hand side is of the same type (that is, typing is not dynamic; if a variable already has a type, it cannot be changed by assignment). In particular, this means that when assigning to a tensor, the expression on the right-hand side must have the same rank as the tensor variable, although the size of each dimension and which indices are up and down may vary.

4.2.1 Declaration

```
vdecl:
    LET ID UNDER LBRACE qual_list_expr { TensorBind(fst $2, snd $2, $5)}
  | LET ID UNDER LBRACE qual_list_expr ASSIGN expr { TensorBindAndAssign(fst $2, snd $2, $5, $7) }
  | LET ID ASSIGN expr { InferredBind(fst $2, snd $2, $4) }
```

To declare a variable, use the keyword `let`. A non-tensor variable can only be declared and assigned a value at the same time. A tensor can be declared and not assigned, though the dimension of each rank must be specified in this case. (Note: these dimensions can be changed later.) The `let` keyword is used to differentiate declaration with assignment from just assignment, since without `let`, assignment would look the same with or without declaring a new variable (which could lead to hard-to-catch bugs). To illustrate declaration,

```
let T_{10,1,17};
let b = 10.5;
let myT = [1,2,3]
```

are all valid declarations: the first declares a rank three tensor `T` (with set dimensions, though these can be changed), the second declares a floating-point number and gives it the value `10.5`, and the third declares a rank 1 tensor of dimension three. The declarations for each data type and details on float and string literals are given in more detail below.

Only one variable can be declared per declaration; that is, each new variable must have its own `let` keyword.

4.2.2 Tensor

```
expr:
  ...
  | LBRACKET ten_list RBRACKET { Tensor(get_line_expr (List.hd $2), $2) }

ten_list: expr COMMA ten_list  { $1::$3 }
  | expr                       { [$1] }
```

Tensors can have any positive rank. There are two ways to declare a tensor.

```
let T = [1, 2, 3];
let  T_{3};
```

In the first,an identifier name is used and simultaneously the tensor must be assigned. This is the same as any other variable declaration. In the second, you can declare a tensor without assigning it(although you can also assign it simultaneously if you'd like). Here, the dimension of each rank must be provided in the indices. This is done by following the identifier with (_) and in braces the demensions separated by commas. The total number of dimension provided is the rank.

```
vdecl: LET ID UNDER LBRACE qual_list_expr           { TensorBind(fst $2, snd $2, $5)}
   | LET ID UNDER LBRACE qual_list_expr ASSIGN expr { TensorBindAndAssign(fst $2, snd $2, $5, $7) }
   | LET ID ASSIGN expr                             { InferredBind(fst $2, snd $2, $4) }

qual_list_expr: expr RBRACE                         { [$1] }
  | expr COMMA qual_list_expr                       { $1::$3 }
```

To define a tensor, one can use nested square brackets, with the number of nested brackets being the same as the rank. Each element of the tensor must either be a float or an integer.
For example, to declare a row vector (rank 1 tensor) $v_i = (1, 2, 3)$, and a matrix(rank 2 tensor) $M = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

```
let v = [1, 2, 3];
let M_{3,3};
M = [[1, 2, 3],
     [0, 1, 0],
     [0, 0, 1]];
```

A tensor's type includes the rank but not the size of each dimension.
Tensor slices can also be accessed. This returns a tensor of lower rank or a float. Any desired slice can be obtained by specifying the index for the desired dimension and dummy indices(any undeclared identifiers) for all other indices. For example, using the declarations from above, `v_{0}` will return 1, M_{0, a} will return [1, 2, 3], and M_{a, 0} will return [1, 0, 0].

4.2.3 Floats

The floating-point type represents rational numbers. Any number that can be expressed as a terminating decimal can be expressed as a float in LaTenS.

A floating-point literal consists of an integer part, a decimal point, a fractional part, and an exponent. Either the integer part or the fractional part (along with the decimal point) may be omitted (but not both), and the exponent is optional and may be optionally signed. This is the same definition of a floating-point number used by ISO C.[2] All floats are signed; the unary minus (−) must be included to signify a negative number while the unary plus (+) is optional for positive numbers for convenience.

The following are some examples of assigned floats:

```
let a = 42.;
let b = 1.7;
let c = -0.8;
let d = .2;
let e = +3.7;
let f = -5.2e+8;
let g = 2e-5;
```

Additionally, every element inside a tensor is also a floating-point number.

4.2.4 Integers

Integers are the standard integers we know and love from ISO C. They consist purely of some non-zero number of adjacent digits from 0-9. Negative integers are preceded by the unary minus (−) while positive numbers can be optionally preceded by the unary plus (+).

The following are some examples:

```
let a = 42;
let b = -5;
```

4.2.5 String Literals

Similar to C, a string literal is a sequence of zero or more characters, digits, and escapable characters enclosed within double quotation marks.[3] As our language is geared towards scientific computation involving tensors, strings are designed to be used solely for debugging. For this reason, certain operations (such as sub-strings) are not allowed. Rather, strings are to be constructed, concatenated, and printed as necessary. String length is also obtainable.

The following are valid escapable characters:

- newline: “\n”
- tab: “\t”
- backslash: “\\”
- single quote ”\'”
- double quote: “\"”

Strings can be concatenated with the ^ operator. (See section 4.2)

To express a string literal, use double quotes:

```
let str = "Hello";
str ^ " world!";
```

will return "Hello world"

4.2.6 Void

The void type is unique: no void variable can exist, and no operators operate on a void type, but a function may have void as a return type. This is simply for the convenience of defining a function that does not return any value.

4.3 Expressions and Operators

There are three possible types of expressions - regular expressions (which can be found in statements and so on); tensor expressions (which can be used inside a tensor immediate–so for example strings are not allowed); and index expressions (which can be used inside tensor indices–so for example tensor immediates are not allowed).

```
expr:
    ILITERAL                { Iliteral(fst $1, snd $1) }
  | SLITERAL                { Sliteral(fst $1, snd $1) }
  | FLITERAL                { Fliteral(fst $1, snd $1) }
  | expr PLUS expr          { Binop(get_line_expr $1, $1, Add, $3) }
  | expr MINUS expr         { Binop(get_line_expr $1, $1, Sub, $3) }
  | expr TIMES expr         { Binop(get_line_expr $1, $1, Mult, $3) }
  | expr DIVIDE expr        { Binop(get_line_expr $1, $1, Div, $3) }
  | expr CONCAT expr        { Binop(get_line_expr $1, $1, Concat, $3) }
  | expr EQ expr            { Binop(get_line_expr $1, $1, Equal, $3) }
  | expr NEQ expr           { Binop(get_line_expr $1, $1, Neq, $3) }
  | expr LT expr            { Binop(get_line_expr $1, $1, Less, $3) }
  | expr LEQ expr           { Binop(get_line_expr $1, $1, Leq, $3) }
  | expr GT expr            { Binop(get_line_expr $1, $1, Greater, $3) }
  | expr GEQ expr           { Binop(get_line_expr $1, $1, Geq, $3) }
  | expr AND expr           { Binop(get_line_expr $1, $1, And, $3) }
  | expr OR expr            { Binop(get_line_expr $1, $1, Or, $3) }
  | MINUS expr %prec NEG    { Unop(get_line_expr $2, Neg, $2) }
  | PLUS expr               { $2 (* Unary plus is redundant; ignore it *) }
  | NOT expr                { Unop(get_line_expr $2, Not, $2) }
  | ID                      { Id(fst $1, snd $1) }
  | vdecl                   { BindExpr(get_line_bind $1, $1) }
  | ID ASSIGN expr          { Assign(fst $1, snd $1, $3) }
  | LPAREN expr RPAREN      { $2 }
  | ID LPAREN actuals_opt RPAREN { Call(fst $1, snd $1, $3) }
  | ID UNDER LBRACE qual_list_expr { TensorId(fst $1, snd $1, $4) }
  | ID UNDER LBRACE qual_list_expr ASSIGN expr { TensorAssign(fst $1, snd $1, $4, $6) }
  | LBRACKET ten_list RBRACKET { Tensor(get_line_expr (List.hd $2), $2) }
```

4.3.1 Unary Operators

_ (underscore)	get element by indices(to index using a predefined scalar, use ())	`let T_a,b = [[1, 2, 3], [3, 4, 5]];a = 2; T_0,0;` returns 1 `T_(a),0;` returns 5
+, −	optional sign. − negates all indices. + is just cosmetic	`let T_a = [1, 2, 3];−T_a;` returns `[−1, −2, −3]`
~	negate the truth value of an expression	~0 is true and ~1 is false

4.3.2 Binary Arithmetic Operators

For a binary operation to be performed on two tensors, it is required that they have the same rank and dimensions.

- +
  Addition. Tensors are added element-wise. When floats and ints are added, the result is a float.
  Examples: `T+T, 5+42.`
- −
  Subtraction. Tensors are subtracted element-wise. When floats and ints are added, the result is a float.
  Examples: `T−U, 42−2.0`
- ∗(overloaded)
  Product.
  - Tensor product.
    Examples: `T_{a}*T_{a}, T_{a, b, c} * T_{a}`
    Each index to be multiplied over must repeat, these must be the same dimension. These are bound dimensions.
  - Scalar product.
    Examples: `4*8.2, a*b, T_{0, 1}*4.`
    A float times an int produces a float.
  - Scalar-Tensor product. Examples: `4*T_{a}, X_{b, c}*T_{0}`
    One must be a scalar, one must be a tensor, order doesn't matter
- ^
  String concatenation. Examples: `"hello,"^ "world!n"`
  Both must be strings
- /
  Scalar division. Examples: `3/5, a/b, T_{0}/T_{1}`
  The divisor must be non-zero.
  When mixing floats and ints, the result is a float.

4.3.3 Assignment Operator

The single equals (=) is designated as the assignment operator. Assignment is mostly by value, but sometimes by reference. All primitive types (i.e. float, int), are assigned by value. All other types (i.e., tensor, string) are assigned by reference. During assignment, the types of the left hand side and right hand side must match (except that ints are promoted to floats when necessary). Assignments are expressions, and return the newly assigned value of the left hand side. For more detailed rules of type checking see the Types section.

4.3.4 Relational Operators

Operator	Description	Example
<	logical less than	`a < b`
<=	logical less than or equal to	`a <= b`
>	logical greater than	`a > b`
>=	logical greater than or equal to	`a >= b`
==	logical equal to	`a == b`
~=	logical not equal to	`a ~= b`

These operators can be used with any type. When used with tensors, the tensors must have equal rank and dimension. The result for tensors is that these operations are applied element-wise and a 1. or 0. is placed in the resulting answer, if the result is true or false, respectively. This is because in LaTenS, 0. (float) and 0 (int) are considered false while every other float/int value is considered true.

4.3.5 Order of Evaluation

```
%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left OR
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS CONCAT
%left TIMES DIVIDE
%right NOT NEG
```

We follow the example of MATLAB when considering operator precedence. Our operators listed from highest to lowest precedence are as follows:

1. Parentheses ()
2. Get element of tensor (`T_{i, j, ...}`)
3. Unary plus and minus (+, −), logical NOT (~)
4. Tensor multiplication (*), scalar multiplication and division (*, \), tensor-scalar multiplication (*)
5. Addition and subtraction (+, −), string concatenation (^)
6. Relational operators: less than (<), less than or equal to (<=), greater than (>), greater than or equal to (>=), equal to (==), not equal to (~=)
7. Logical AND (&&) and OR (||)
8. Assignment (=)

The assignment operator, the logical NOT operator, the unary minus, and the underscore associate from the right. All other operators are left-associative.

4.4 Structure

4.4.1 Statements

```
stmt:
    comment expr SEMI                               { Expr (List.rev($1), $2) }
  | comment RETURN SEMI                             { Return (List.rev($1), Noexpr) }
  | comment RETURN expr SEMI                        { Return(List.rev($1), $3) }
  | comment LBRACE stmt_list RBRACE                 { Block(List.rev($1), List.rev($3)) }
  | comment IF LPAREN expr RPAREN stmt %prec NOELSE { If(List.rev($1), $4, $6, Block([], [])) }
  | comment IF LPAREN expr RPAREN stmt ELSE stmt    { If(List.rev($1), $4, $6, $8) }
  | comment FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN stmt { For(List.rev($1), $4, $6,
      $8, $10) }
  | comment WHILE LPAREN expr RPAREN stmt           { While(List.rev($1), $4, $6) }
```

A statement is either an expression or some compound statement of expressions. An expression evaluates to a value, meanwhile a compound statement of expressions has no associated value and only defines a side effect. Every statement must be terminated with a semicolon or enclosed in braces. Some examples of expressions are:

```
[2, 4, 5];
```

The above value expression has no side effect, but evaluates to an immediate value of $[2, 4, 5]$, a single dimensional tensor (a vector).

```
let V = [2, 4, 5];
```

The above expression defines a vector variable (1-tensor) and assigns it the immediate value of $[2, 4, 5]$. The above value expression has both a side effect and an evaluation of $[2, 4, 5]$.

```
s = [2, 4, 5] * [2, 4, 5];
```

The above value expression performs tensor multiplication between two vectors, $[2, 4, 5]$ and $[2, 4, 5]$, and assigns the resultant value to the variable s. This will assume that the first index of each is the one summed over, for vectors this looks like a dot product. The value expression also evaluates to the newly assigned value of s, which in this case is 45.

```
v = foo([2, 4, 5]);
```

The above value expression calls the function foo with a single argument $[2, 4, 5]$. The above assumes that foo is a function taking a single argument, a 1-tensor, and returns a value. The syntax and semantics of functions will be covered in a later section.

Compound statements that do not evaluate to a value are typically found in code flow constructs, and allow the flexibility of evaluating to no value in situations where there is no suitable candidate evaluation value. Some common void expressions are:

```
if (evalexpr)
{
    stmt1
    stmt2
    ...
    stmtn
```

```
};
```

The entire *if* block above is a statement that evaluates to no value, this means the following is illegal:

```
v = if (evalexpr)
    {
    ...
    };
```

The above is illegal because *void* is not a value, but rather a guarantee that *no* value exists. Another common void expression is found in loops:

```
for(beginstmt; evalexpr; iterstmt)
{
    stmt1
    stmt2
    ...
    stmtn
};
```

Function evaluations can void. If so, the following would be illegal:

```
v = bar([2, 4, 5]);
```

4.4.2 Expression

An expression is one of assignment, unary or binary operators applied to expressions, or a *non-void* function evaluation.

4.4.3 Statement

A Statement is one of statement sequences, expression, return, if-then-else, for loop, or while loop.

4.4.4 Block Statement

A block statement is a void expression, which comprises of zero or more statements. The syntax for a block statement is as follows:

```
{
    stmt1;
    stmt2;
    ...
    stmtn;
}
```

4.4.5 True and False

Boolean types do not exist in LaTenS. There are analogues, however. In particular, only the float 0. and the int 0 evaluate to false. All other float and int values are considered to be true.

4.4.6 If-then-else

An if-then-else comes in two following forms:

```
if(evalexpr)
stmt1
```

```
if(evalexpr)
stmt1
else
stmt2
```

The semantics of the above is that if evalexpr is evaluated to *True*, then the if branch is executed. If an else follows the if, then it is executed only if evalexpr evaluates to *False*. The execution of *stmt1* and *stmt2* is mutually exclusive.

4.4.7 Loops

The syntax of for loop in LaTenS is as follows:

```
for(beginstmt; evalexpr; iterstmt)
stmt
```

The semantics of the above is to execute *beginstmt* in the prologue, and continuing to execute the loop body while *evalexpr* evaluates to *True*. To be more specific the loop body consists of *stmt* followed by *iterstmt*. Prior to any execution of the loop body, *evalexpr* must be evaluated and checked to be True.
The syntax for while loops is as follows:

```
while(evalexpr)
stmt
```

The semantics is to execute *stmt* as long as *evalexpr* evaluates to True.

4.4.8 Function definition

```
fdecl: typ ID LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE
       { { typ = $1; fname = $2; formals = $4;
           body = List.rev $7 } }
```

The return type may either be *void*, or some other concrete type.

4.4.9 Scoping

Functions and expression sequences intimately interact with *scope*. Abstractly, a *scope* is a set of bindings between identifiers and their underlying meaning. In particular each expression sequence defines a container of mappings of identifiers to their meaning (a scope).
During the evaluation of an expression sequence each *assignment* statement adds a mapping from the identifier to its meaning in the container corresponding to the current scope. Because expression sequences can nest, and there are many such containers each having possibly conflicting mappings. Therefore there are strict rules for evaluating the meaning of an identifier based on commonly accepted scoping rules.
Define the innermost scope as the mapping belonging to the currently executing statement block. Define innermost-0 as innermost. Define innermost$-i$ as the mapping belonging to an statement block which

contains an statement block using innermost$-i+1$ as its scope. Define outermost as any scope not contained in any other scope. Given the above definition the following rules are followed for mapping:

```
map(id)
{
    let i = 0
    while(innermost-i != outermost)
    {
        if(innermost-i.contains(id))
        {
            return innermost-i[id];
        }
        i++;
    }
}
```

Function definitions also subtly interact with scope in a slightly different manner. In particular, a function definition adds the function to the *outermost* scope. Furthermore, this addition is performed prior to the execution of the statement associated with the function definition. This allows a function to be in its expression sequence's scope. This is necessary for recursion. Note the above rules do not allow forward declarations. In particular mutually recursive functions are not allowed in LaTenS.

4.4.10 Program

```
program: decls EOF { $1 }
```

A program is a sequence of function definitions ending with a specially named function called *main*:

```
returntype1 functionname(argtype11 arg11, argtype21 arg21, ..., argtypen1 argn1)
stmtlist1

returntype2 functionname(argtype12 arg12, argtype22 arg22, ..., argtypen2 argn2)
stmtlist2

...

returntypem functionname(argtype1m arg1m, argtype2m arg2m, ..., argtypenm argnm)
stmtlist3

void main()
stmtlistmain
```

Every program must contain a main function that takes no formal parameters. This main function is the entry point of the LaTenS program. Program execution begins at the first statement in this main function. In other words, the executed main function takes no arguments and can return any type.

4.5 Standard Library

A standard library is always included when compiling LaTenS code. It includes the following functions

- file operations:
  - fopen: Takes two strings, a filename and a mode. Returns an integer file pointer. Mode can be:

        * "r" - Opens a file for reading. The file must exist.
        * "w" - Creates an empty file for writing. If a file with the same name already exists, its content is erased and the file is considered as a new empty file.
        * "a" - Appends to a file. Writing operations, append data at the end of the file. The file is created if it does not exist.
        * "r+" - Opens a file to update both reading and writing. The file must exist.
        * "w+" - Creates an empty file for both reading and writing.
        * "a+" - Opens a file for reading and appending.
    - fclose: Takes a string filename. Returns an integer which is 0 if file closed successfully.
    - fread: Takes two integers, the number of bytes to be read and the file pointer, Returns the string read.
    - fwrite: Takes the string to be written and an integer file pointer. Returns the number of bytes written.
- print: Takes any datatype. Returns the number of characters written.
- strlen: Takes a string. Returns the length of the given string.
- dim: Takes a tensor and an integer where the integer is the index that is being requested. Returns the dimension of the given index of the given tensor.
- casting:
    - itof: Takes an integer. Returns the float of that integer.
    - ftoi: Takes a float. Returns the integer of that float.
    - itos: Takes an integer. Returns the string of that integer.
    - stoi: Takes a string. Returns the integer of that string.
    - ftos: Takes a float. Returns the string of that float.
    - stof: Takes a string. Returns the float of that string.
- math:
    - sqrt: Takes a float. Returns the square root of that float
    - pow: Takes a float and an integer. Returns the float raised to the integer power.
    - abs: Takes a float. Returns the absolute value of that float.
    - transpose: Takes a rank-2 tensor. Returns the transpose of that tensor.
    - min: Takes a rank-1 tensor. Returns the minimum element
    - minInd: Takes a rank-1 tensor. Returns the index of minimum element
    - max: Takes a rank-1 tensor. Returns the maximum element
    - max: Takes a rank-1 tensor. Returns the index of maximum element

5 Project Plan

5.1 Planning Process

5.1.1 Overall Plan

As a group of four, we began by divvying up the the project roles of manager, language architect, system architect, and tester as areas of general responsibility even though they wouldn't all be needed at first. Our

plan was to decide on some features that would define our language, write a well-thought out LRM, and use this to create a scanner/parser/abstract syntax tree (AST) and as many test cases as we could predict together. At first we intended this to be the point at which we could split up and begin to own our own features but realized for work on code generation to get done, we needed some idea of what our semantically-checked AST (SAST) would look like. We had to sit back down, make a very high-level semantic checker, and draft a rudimentary SAST but then we moved on to individually picking the features we wanted to implement as parts of semantic checking and code generation. Whenever we implemented a feature for which we didn't yet have tests, we also added test cases for it.

5.1.2 Specification Plan

Our Language Reference Manual had clear specifications of our syntax and semantics. When writing our LRM, we put a lot of thought into keeping it feasible and well-defined, so we hoped we wouldn't have to change too much of our initial specifications throughout the semester. It turned out we had put a little too much effort into it. Our parser caught a lot of semantic errors that we ultimately preferred to catch in our semantic checker for two reasons: because it was easier to handle semantic-checking (OCaml) exceptions than parser (OCamlyacc) exceptions and more importantly, because the more bloated our parser became, the harder it was to modify it as needed. Ultimately, this led to a major revamping of our parser near the end of November, in which we added a bunch of new features and cleaned up a large amount of bloat and inefficiency.

5.1.3 Development Plan

We moved through development in the what seemed to be the logical order. We began with a scanner, parser, and AST, which were done together with our LRM. From there we designed a rudimentary SAST. Three of us chose to begin work on different semantic checks while our fourth started implementing basic features in code generation. Our rudimentary SAST got modified a lot as a result of work on both semantic checking and code generation. As our semantic checker began to take shape, more of us moved to code generation and ultimately every teammate had their hands in every file we wrote.

5.1.4 Testing Plan

We wrote our original test suite together with our scanner/parser and LRM. From then on, it was the job of whoever implemented a feature to come up with suitable tests for it and our tester to ensure that that was happening.

5.2 Style Guide

We attempted neat OCaml code with spacing, indenting, and new lines but didn't have any explicit rules. At times, spacing was complicated since two of our teammates coded on Macs in emacs and 2 coded on Ubuntu in Vim. When code was particularly unreadable, it was explained to the group and at times, the manager went through the code and cleaned it up.

5.3 Roles and Responsibilities

Our original scanner, parser, and AST were group efforts. From there, skeleton responsibilities were:

- Daniel: semantic checking (semant.ml) w/ Elsbeth and Eliana, exceptions (exceptions.ml)
- Elsbeth: semantic checking (semant.ml) w/ Daniel and Eliana, test suite (tests/)
- Eliana: semantic checking (semant.ml) w/ Elsbeth and Doni, SAST (sast.ml)

- Mohit: code generation (codegen.ml)

After the skeletons were in place, we created a list of features to implement within a Github issue log. Every member of the group was charged with "claiming" a feature by placing his or her name next to it on the list, implementing it, moving it to the "done" category, and repeating the process. As such, we kept a record of who implemented what and in when. This required every member of the group to write code in our semantic checker and code generator and also modify the parser, AST, and SAST at times.
Some highlights are:

- Daniel:
  - updated Makefile + scripts (testall, script for counting number of passing and failing tests) + maintaining the git repo and managing tasks
  - a large number of semantic checks + semant side of built-in C functions
  - tool for adding LaTenS standard library to generated code + half of the standard library w/ Eliana
  - error handling: Exceptions.ml + ocamlyacc/lex error catching as well as getting semant error messages to print line numbers
  - pretty printing code into LaTeX w/ Eliana
  - keeping codebase neat except for codegen
- Eliana:
  - LaTenS standard library w/ Daniel
  - some of codegen, including promotion and many of our built-in C functions w/ Elsbeth
  - GR Demo, Cows demo
  - pretty printing code into LaTeX w/ Daniel
  - semantic checking w/ Daniel, Elsbeth
  - semantic checking of tensors w/ Elsbeth
  - Pretty printing for the SAST
  - Defined language syntax and function
- Elsbeth:
  - Implemented smaller code generation features, including promotion and many of our built-in C functions w/ Eliana
  - Perceptron demo, basic code demo
  - Pretty printing for the AST
  - Semantic checking of tensors w/ Eliana
  - Test suite, test script
  - Semantic checking w/ Eliana, Daniel
  - Final presentation slides
- Mohit:
  - Transformed Micro-C codegen to LaTenS compatible barebones codegen.
  - Fixed existing Micro-C features such as functions, loops, if-else statements, and integer operations
  - Implemented Tensor specific code generation including Tensor slicing, Tensor multiplication, Tensor allocation, and Tensor binary operations.

5.4 Timeline

Sep 28: Project Proposal

Oct 26: Language Reference Manual

Nov 17: Major semant update, lots of languge specific checks added and relationship to MicroC semant ended

Nov 20: Hello World! compiles and runs

Dec 5: Pretty-printing into LaTeX finished

Dec 5: Major update to parse tensors in a logical manner

Dec 12: Tensors finished in everything but codegen

Dec 19 Present! Code demo!

Dec 20 Submit Final Report

5.5 Dev Environment

We wrote our compiler in OCaml 4.0.1 with OCamlyacc and OCamllex. Our code generated LLVM-IR for llvm-3.8 and linked to glibc functions. Our project has an up-to-date Makefile and test cases are run and analyzed via shell scripts. We coded in a mix of Sublimetext2, Vim, and Emacs.

5.6 Git Logs

```
commit a0e1e20740d6c4ac6caa070001cf07f09e266588
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Tue Dec 20 19:13:03 2016 -0500

    Removed some useless files

commit 64516f70f0275e524d8c9f96a40cacaeb6f218e7
Author: Elsbeth Turcan <elturc@gwmail.gwu.edu>
Date:   Mon Dec 19 16:30:28 2016 -0500

    new powerpoint

commit 88610f63d9c60074d98bb98ac578449263b6a065
Merge: 023ed31 5414b70
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 15:57:24 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 023ed3187beb4fa631664259fa7135be9e54f6c5
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 15:57:19 2016 -0500

    presentation with more slide

commit 5414b706c7da4c7b8fbad17c38666860ef150fab
```

```
Author: dyschwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 15:32:34 2016 -0500

    semant neatified

commit 5ed267b43f1d9cb61a9e6154e8356204a2b3a5f1
Merge: fa64c3c bc577c2
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 15:29:08 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit fa64c3c303181c3be3b808871482c66da9d0dfe7
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 15:28:45 2016 -0500

    finished demo0

commit bc577c2e00a4b952bce6812aabac57a4dbe6a1a9
Merge: c35fd06 ef64bc9
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 15:18:47 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit c35fd06ade33e710c6564528ac7fbb18ab707322
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 15:18:38 2016 -0500

    latex demo

commit ef64bc97f116ed225e213d6677ee8190d3e57b7e
Author: dyschwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 15:18:35 2016 -0500

    updated presentation

commit c2ac7e7e24f79e0a216e7294b252442ce8678b43
Merge: 8d01995 9fe2432
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 15:05:40 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 8d0199517b50acd72173f01b004dd2b61e41591d
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 15:04:52 2016 -0500

    fixed minInd, maxInd

commit 9fe2432adfd588dd03d89a843f4adf99030db608
```

```
Author: dyschwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 15:04:06 2016 -0500

    abs test fixed

commit 36e1ba9ddd23ac356e2adf950c84d3d7bba76ae5
Merge: 5c3d57b de0370d
Author: dyschwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 14:59:26 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 5c3d57ba297dc37ca7964e7259dc93b5080aa9e7
Author: dyschwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 14:59:19 2016 -0500

    tests updated

commit de0370dc1eda259e35761b4aafd5a3dcef700f8f
Merge: 6958f00 96d7a37
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 14:53:30 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 6958f0087948f353a88979d24261ae0d3937afa8
Merge: a6e78b7 b9ecfa5
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 14:52:51 2016 -0500

    std lib tests

commit 96d7a376491cd73fbabfbb600c78426234c2e44e
Author: dyschwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 14:51:35 2016 -0500

    test for overloading

commit a6e78b7b20fb486e01901a6814f72f6b80049625
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 14:42:58 2016 -0500

    demo

commit b9ecfa5241ab5f086d054e6660bdafd4f340602c
Author: dyschwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 14:41:17 2016 -0500

    tests for stdlib added

commit dfd8403e2b67928f96589ba4cca6991221ad63e8
```

```
Merge: 4944e5e a6098e5
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 14:30:02 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 4944e5e5c390162fea1ae4974e9a7a6586c6d49d
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 14:29:54 2016 -0500

    absolute value

commit a6098e549eb5c735532ddbb17687f83b69f9cb55
Merge: b0b3d79 bf59851
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 14:26:21 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit b0b3d79296010b38c2e6153d4eb8a5c236df194d
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 14:26:12 2016 -0500

    Added simple syntax test

commit bf598515786eefd7b0535ddec2487a75194f079e
Merge: 6350615 d0ccc29
Author: dyschwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 14:23:23 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 63506152df29b5a050c6a786a97688630e3bafce
Author: dyschwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 14:23:17 2016 -0500

    library errors fixed

commit d0ccc29f0480184fd9fba6cfd9856a23ae7035ee
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 14:21:30 2016 -0500

    strlen

commit 19ffd51e96eacb5c71a6a41aad5da3ddaf53f98d
Merge: a63caf3 a61bd67
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 14:14:51 2016 -0500

    merged
```

```
commit a63caf34aa800c6b505707c055102ee23657a1eb
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 14:13:54 2016 -0500

    min/max

commit a61bd67dddf7f0805cc4448e1567f0553cf20ec9
Author: dyschwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 14:09:18 2016 -0500

    pretty_print latex neat-ified, presentation updated

commit d6cd3af3e2edebb653626d0c5dcce8acb0465221
Merge: 007b0c9 100a901
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 13:49:19 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 007b0c91643f28214cdae2eb4929763a84655614
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 13:49:14 2016 -0500

    transpose in stdlib

commit 100a9010794ac39382a1404c35f1f9fcf6d823ed
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 13:40:45 2016 -0500

    5x16 perceptron

commit 8422f9c7a6cba12a32bda9bc5a133b3448bd3084
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 12:58:33 2016 -0500

    demo2 works

commit 1ce6180d72e5f0f2c01275410ec71937cf585b8b
Merge: 18313d2 f31f971
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 12:45:19 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 18313d26f7e0db881e8c6049c4b7c2f4987fa214
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 12:45:08 2016 -0500

    Perceptron working!

commit f31f971601be8b2e4e1bdcd968fc6d88123fe088
```

```
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 12:42:50 2016 -0500

    demosos

commit a5352be1e252b3e92060a29bcc4179cd8b2b5876
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 12:26:39 2016 -0500

    demo hopefully works

commit eed02ab1a69fee3d711f03007f31f9fd6c104215
Merge: 1f31668 8c8c4b3
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 12:13:02 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 1f31668067779b87fcdb75da2c2d72ef363b2874
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 12:12:43 2016 -0500

    demos

commit 8c8c4b328dcdc9635c837a4cbf68e7a884d05212
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 12:08:18 2016 -0500

    Perceptron example

commit 084acea8100025171119b7a8698a27d0f58ab7ba
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 11:56:48 2016 -0500

    demo stufffff

commit b5952885e98c0e3e3a10f46c838d7599b189170d
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 11:52:50 2016 -0500

    more demo tests

commit 80fc5b174a594962c4a3952c35712ff4393c3415
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 11:49:21 2016 -0500

    more demo checks

commit 171253af915f33077a0b33932e40e0e02371b990
Merge: 09215f4 1dfbc65
Author: ElianaRose <erw2138@barnard.edu>
```

```
Date:   Mon Dec 19 11:41:41 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 09215f45b1ea545fcefdfa0dda9724bf53cdcf3c
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 11:40:01 2016 -0500

    demos/demo2.tens

commit 1dfbc65d04d5b4c7e6fcfe4a2549171220d330f9
Author: dyschwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 11:33:37 2016 -0500

    yay, now my commits count towards my total on github (added my git name to my local computer). also

commit 6128c8cef5969322234d81f82ce619adbaf61463
Merge: 93ec317 64d7c85
Author: dyschwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 11:27:19 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 93ec3176b074ebf1cc537ee3ed16700c10cb967c
Author: dyschwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 11:27:14 2016 -0500

    latex comments added to Ast.stmt, demo1 nice-ified

commit 64d7c8527ca5ae2149f20192a0ea2b683a32b20b
Merge: dda67ee 685d91f
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 11:24:49 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit dda67ee6386d15ce9ea66c023a30c02d20092a97
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 11:24:41 2016 -0500

    more demo edits

commit 685d91fcf7b89bd0de070911883420de4a7b67a3
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 11:21:38 2016 -0500

    Fixed powe

commit 33822b51e6bcd860825c49f9aa9b5252fa840260
Merge: 3df758c 66c5a2f
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
```

```
Date:   Mon Dec 19 11:16:11 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 66c5a2fe6b2bc61a3e684d3c36db6e2e29cb95ea
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 11:15:50 2016 -0500

    power test

commit 3df758c6901a5f4d7eb0e26f3b0374109ec3ab06
Merge: d8c37af 671e764
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 11:05:47 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 671e76401899851f10d30684bc584c72c86ae95a
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 11:05:05 2016 -0500

    demo2+power

commit d8c37afd766413f634e82e7560e67065c2dd7dd7
Merge: 0eb8de9 b9705ff
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 10:50:18 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 0eb8de9f9db1a2af67df4fda209c69485a1018a1
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 10:49:30 2016 -0500

    oops. here's the rest of the tests

commit b9705ff60beb2fb45eaa69dfd90fc5e2d84e4c4b
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 10:30:49 2016 -0500

    neatified makefile

commit 39db1cafe96f9c83548fe05bf78f995f78fd0c88
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 10:28:34 2016 -0500

    warnings in codegen fixed, added test for float syntax, properly clean up intermediate file in later

commit f3b55119a3c41fb16664b71bd24ee2a8e45ce2e9
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 10:06:45 2016 -0500
```

```
    compiler supports just giving filename and no flag

commit d71d2f623a53b7e3b413a951b03c9242e7e9480c
Merge: bfefd57 16bb955
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 09:57:29 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit bfefd570fc6489ef6ed042f4f70484eab1253fba
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 09:57:26 2016 -0500

    fixed compiler to work with test cases

commit 16bb9557f59c3c3087424b2d9bf1a674fd4503cf
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 09:55:08 2016 -0500

    sqrt

commit 37e6e8b4bb8ea604aa04f60900a3dc8725e0674a
Merge: 4b3372b 3d3e652
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 09:46:24 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 3d3e6521f521029228bdf1369f9b5bb5362de4c8
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 09:46:16 2016 -0500

    improved compiler to take 1 or 2 args

commit 4b3372b6e491985609de99dba1e602a8450c53e3
Merge: dcb474d 7f6de75
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 09:42:32 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit dcb474dad56678feb07fc79eac4cf38c1e5c798b
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 09:42:28 2016 -0500

    more demos

commit 7f6de752a15de256aebdc7791daf68caf55ee569
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 09:31:29 2016 -0500
```

```
    stdlib functionality added

commit 9072906b9b5610bf7e71ea68c253c55d4d032e40
Merge: 64be97c 33ae60b
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 01:05:05 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 64be97c73fd1e7dd950f19b43d535d09de76fefc
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 19 01:05:01 2016 -0500

    first demo :)

commit 33ae60b515cf595b4278aec0be8486341509cd43
Author: Elsbeth Turcan <elturc@gwmail.gwu.edu>
Date:   Mon Dec 19 01:00:59 2016 -0500

    Added presentation by Doni's request

commit 6af996c1010caa339b98115882d4b26081c3b905
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 00:38:41 2016 -0500

    semant warnings taken out. should be pretty much finalized for presentation

commit f0da241e3afed2c2c6b86b172e759493df02b4da
Merge: 0aec399 0d0a0bc
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 00:17:53 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 0aec39905d88d431eed15fe02b981814c102d5f5
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 00:17:41 2016 -0500

    removed bad tests

commit 0d0a0bc7273a7f55ead00efef252b78fd696b6a3
Merge: f6c0c64 2bc468d
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 00:16:31 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit f6c0c6499dd2ca31c24909ff1b8ff857ff9ec519
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 00:16:19 2016 -0500
```

```
    Fixed dot product test

commit 2bc468d5309ff713521ddf932d7fb78ccfec62e2
Merge: 7cde2b1 6ac8559
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 00:12:10 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 7cde2b19ac12bae063e53fe3d18a18d7dcccbfaa
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 19 00:12:08 2016 -0500

    more tests working

commit 6ac855969f21ce19ab0625c9bb335ea159f85038
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 00:09:43 2016 -0500

    Fixed test-func2 out file

commit c04185b677f8c4225926ccdb352c82c94406fdc8
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 19 00:05:42 2016 -0500

    dot product fix in progress

commit e1f6da694fa67bbc2ee697282b3b3bb733be4dfd
Merge: 09650de d748a6c
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 23:50:38 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 09650def20627ec2b818d1c756e70f3345719e1d
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 23:50:30 2016 -0500

    more test fixes

commit d748a6c320963a90d05f220f4ff5825526901506
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 23:49:49 2016 -0500

    took debug printing out of semant. sorry guys

commit a7ab0c614e45510d90a0269fc991180622ac5418
Merge: d73a8f7 84baf65
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 23:42:43 2016 -0500
```

```
    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit d73a8f79497fe8bcbd9ca639992d0802f6df82bd
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 23:42:39 2016 -0500

    fixed return checking in semant - doesn't break tensors anymore

commit 84baf6592651abd829a04a9b503a21d1c10ff943
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 23:39:16 2016 -0500

    0-tensor not propogating

commit 24ac422af8a3c67bd12ce6cb525191d3293f8382
Merge: d118d6e 3a00ecf
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 23:26:49 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit d118d6eaa13c2d75ea10266fb228218d8a837d99
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 23:26:45 2016 -0500

    more tensor tests

commit 3a00ecf7d567afb6f5d10df3afb44c5680de7dd9
Merge: 0b7603d 0b8eaa8
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 18 23:25:40 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 0b7603d35cc4792debc884b97f9385243ad0bec2
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 18 23:25:33 2016 -0500

    Added promotion in an assign also

commit 0b8eaa863bbfdf003c44cb7ecbea47cd1c2831cd
Merge: 3dd4af3 faa96a9
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 23:21:58 2016 -0500

    Merge branch 'master' of github.com:dyschwartz/LaTenS

commit 3dd4af335f0864227919c15d6520f9ef1f615a94
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 23:21:51 2016 -0500
```

```
    stuf

commit faa96a949cd326d162f20e4cc17fe9c01fb6df27
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 23:16:48 2016 -0500

    fixed while test

commit 60daefb8b4d0fd91802fbcaab12019bdf1ba2b8a
Merge: eac7540 26795bd
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 23:11:11 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit eac75400b0aa53e6b8bf68515f53d515e9bcf971
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 23:11:09 2016 -0500

    fixed tensor mult test

commit 26795bd17e971e7a01d10a1cfbb9333604b940d5
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 18 23:10:32 2016 -0500

    Fixed string test

commit 3de80e98c85ddb0ecf108006bf4b3330225e30a1
Merge: bc54bba e9d4ce6
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 23:02:11 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit bc54bbaef2a047e9cb21c1d36205207c4360e882
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 23:02:09 2016 -0500

    script to count number of tests passing and failure

commit e9d4ce63d139ad2a0b2cb1c8d71f26a6c3414eb4
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 18 23:01:30 2016 -0500

    Fixed another return type error

commit 3df12e764ef30b88912b141ba5f8fcf5388e754d
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 18 22:58:05 2016 -0500
```

```
    PROMOTION!

commit 348ed7fc80c7ba64a66361cd34ab6fe1b79ce3eb
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 22:45:30 2016 -0500

    codegen fix

commit 2d3cd9c781812248dc3ab9df476598a12758d240
Merge: 3d10216 1bbf6f8
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 22:42:00 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 3d10216a2483859c1ef74841eb70ac21147e837d
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 22:41:57 2016 -0500

    finally check correctly to see if function blocks end with return statements while other blocks don

commit 1bbf6f8c5110082ab3042230600749e6dcf0635b
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 22:41:01 2016 -0500

    promotion

commit b0bc5f3b9ad51dce621f3d9715b0e4712f0220be
Merge: ece6fc7 9ccef02
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 22:21:59 2016 -0500

    Merge branch 'master' of github.com:dyschwartz/LaTenS

commit ece6fc7f07e1c13d112baf456d0a7d8dc5161da9
Merge: fc10090 0077942
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 22:21:52 2016 -0500

    Merge branch 'master' of github.com:dyschwartz/LaTenS

commit 9ccef02b29a84d87c368d3eece4f914469660866
Merge: 34e5647 0077942
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 22:21:49 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit fc10090ca5c89115000a458bf054d2da64ca7499
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 22:21:43 2016 -0500
```

```
    tmult bug fixed

commit 007794244bcee541417fc1fb25c78f8cb8a0fa3f
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 18 22:21:18 2016 -0500

    Added more tests

commit 34e56477f8a0b105fb22f67482682787502732ce
Merge: 888edbb 81f8a12
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 22:20:56 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 888edbb8f1ddff68b1ddb42f0b352eb9c5a1d904
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 22:20:52 2016 -0500

    more tensor tests

commit 8ecf2791db8e68af619445e48213e6948826f4ad
Merge: f8c481d 81f8a12
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 18 22:08:26 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 81f8a123e79c835514789039b53474ce6db2c480
Merge: 5894663 23ebc24
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 22:08:04 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 589466379f09de1f964a6f5abb2f3dd117bf872e
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 22:08:00 2016 -0500

    more test cases. semant closer to fixing internal scoping error

commit f8c481dce2940f221087ced89226bdba7b463779
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 18 22:07:54 2016 -0500

    Added more tests

commit 23ebc2472b6316d8acf45d02c79017a769a7432a
Merge: e0d7745 f95dd74
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
```

```
Date:   Sun Dec 18 21:58:35 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit e0d774549214ff13234a797ebe091aa920ee578c
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 18 21:58:27 2016 -0500

    Added more tests

commit f95dd74f736c1e408b59c1ca253920cc546dd982
Merge: 91c5551 46133bf
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 21:46:37 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 91c555131124a85b95ac3f8993530eb4edec02a0
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 21:46:35 2016 -0500

    tmult1 test added

commit 46133bf935373256c55ee7d3157f07b871840bae
Merge: 8b6f099 a95d45f
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 21:33:24 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit a95d45fa97e3e74aad5df319acfab293a651730b
Merge: 1badf3b 39af5ea
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 18 21:31:29 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 8b6f0994a122921a774e025c7c86a990ada8c67a
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 21:31:28 2016 -0500

    more test cases, better error handling on return statements

commit 1badf3b37fee4de0270083720474c7169f73af5d
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 18 21:31:18 2016 -0500

    removed some warnings

commit 39af5ea4c4ae6474e2897fe28a8cd0285747ec42
Author: ElianaRose <erw2138@barnard.edu>
```

```
Date:   Sun Dec 18 21:29:39 2016 -0500

    multiplication gets correct type

commit 95642db5ca4610f110cdc89b8881cf236c0fad82
Merge: d680322 9545a57
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 20:51:33 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit d680322ce4f374c2aa55eccfc176effb71aa83b5
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 20:51:29 2016 -0500

    more test cases. semant catches missing return statements

commit 9545a574deba0b85857df3d273183c4fa3364946
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 20:48:57 2016 -0500

    printf tensor works

commit 4d75c89e2d763c7b9232fb4a24dad6cd93d8b2c6
Merge: 8ac53bc 2d0d262
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 18 20:28:28 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 8ac53bcc07169b9eeeec70305006081a72fb2853
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 18 20:28:14 2016 -0500

    Tensor print in semant, added a test

commit 2d0d262419ebf253c734548d796ff2f9e3908517
Merge: a80563d 8ded60e
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 20:19:19 2016 -0500

    Merge branch 'master' of github.com:dyschwartz/LaTenS

commit a80563d2b1b9fa546f79709c64a23bb6f5bf4c48
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 20:19:07 2016 -0500

    printf codegen

commit 8ded60e36b055d1f8db36b31621334e9bc40e7ac
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
```

```
Date:   Sun Dec 18 20:02:42 2016 -0500

    Added missing .out file

commit f075683f73a14742a5f2f50e7f3a475398629da5
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 19:48:52 2016 -0500

    more test cases

commit 7d43f94ac7f7dd015ed536c344eac4fe81010b42
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 18 19:44:00 2016 -0500

    Fixed tensor sub2

commit 393d162c2ab38d86f04db37b0b9eba9339683267
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 19:34:42 2016 -0500

    while tests, tensor add test working

commit c23bbae8c027e3ecd0dfd99dc69722147d748d20
Merge: 8f9a158 196b932
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 19:18:13 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 8f9a158370ad424a759122a1c1e926afd5f010fd
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 19:18:10 2016 -0500

    semant binary boolean operators range expanded to handle all types

commit 196b932a81650687d4e8002d03231407db37d904
Merge: 45594cc 0abca39
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 18 19:06:51 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 45594cc147d7a01be35f6474fb24cf7afa9a66ff
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 18 19:06:37 2016 -0500

    File I/O in codegen

commit 0abca39615e462575ff4b4ff6efca40a09715f58
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 18:55:19 2016 -0500
```

```
    new test case solutions added

commit 0138235c7d4e2d394d95c934308e196f8e56900e
Merge: 8e5604e 2c232ac
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 18:41:59 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 8e5604ecfd5de68ddad8d24e7f3cfbaabab1c93a
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 18:41:52 2016 -0500

    correct scoping and test added

commit 2c232acfbe4184b21123c2355c49ca8b28373a09
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 18:26:42 2016 -0500

    Tensor mult works

commit 8a6d4865adb00bc8a766096681de737f2d15ae66
Merge: fb35754 94ffa5e
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 17:56:37 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit fb35754e075c4b606fb94cc1209925afe0b3ac82
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 17:56:34 2016 -0500

    fopen etc.

commit 94ffa5e0e4d8e63167429ca118aa07ef6590f7b7
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 17:56:32 2016 -0500

    codegen tmult bug fixes

commit 9eec40ba41ad12b8ca7123b16fec150c3162c0b9
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 16:40:33 2016 -0500

    builder -> builder_ref fixes

commit dae29fb76081ed004e1bc80d8b5e88d0f98a5ed5
Merge: c6862ae 2442af4
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 16:38:26 2016 -0500
```

```
    Merge branch 'master' of github.com:dyschwartz/LaTenS

commit c6862ae34c7fbbea30bea7686c47984d0aafebf7
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 16:38:20 2016 -0500

    ITS OVER 1000git add codegen.mlgit add codegen.mlgit add codegen.ml

commit 2442af474213e1707c588ff30b03bfe270cb501b
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 15:29:46 2016 -0500

    fixed loops and added tests

commit 3c27d62794597e967178f8ce16945dfa18f0a9fc
Merge: 940729d e992275
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 14:05:04 2016 -0500

    Merge branch 'master' of github.com:dyschwartz/LaTenS

commit 940729d610cda5c4c49fb57c5abd7bf3889d6402
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 14:04:56 2016 -0500

    LaTeNs > DICE

commit e992275e506428a765a0623b56c4dc6abc75e1da
Merge: 1c9715f 99f9661
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 13:31:01 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 1c9715f5663c50c563783a1e3f9d487e5116ed60
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 13:30:59 2016 -0500

    fixed float-ops and fail-and

commit 99f96617e40bdd8ed257460a6c9dca94f92323ea
Merge: 61fef05 55c6f6f
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 13:28:55 2016 -0500

    Merge branch 'master' of github.com:dyschwartz/LaTenS

commit 61fef05a17f590ccc15400c38ce78ba44c2cf218
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 13:28:50 2016 -0500
```

```
    bug fixes, tensor mult work on the way

commit 55c6f6fa325142d399bc6e4ec2542e1da676c894
Merge: 3a62638 4185a73
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 12:23:17 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 3a62638ff6e70f45f36217738da9dc181c6dd085
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 12:23:13 2016 -0500

    fixed if tests

commit 4185a73eff29b0ebf187712e7de6b50a98423c53
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 18 11:58:03 2016 -0500

    added line numbers to some more exceptions

commit a985acdf3bf9f73e22f9450576fa8a2e888e49e4
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 18 11:22:14 2016 -0500

    no more tmult

commit 4b02347c7659673783593f14697b6e9979e2b099
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 18 11:01:30 2016 -0500

    iterator bug

commit 47cf9d9ea7d254f2eba7e9fd4a9a7bf9f6037397
Merge: 2186600 34f49b2
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sat Dec 17 23:16:16 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 2186600ff22eca47bc05226a4403b36a9212d9a2
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sat Dec 17 23:16:00 2016 -0500

    more tests. Also semant doesn't allow for logical comparison of tensors

commit 34f49b2b73085eaabb91327d0e0c94cc66cbe696
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sat Dec 17 22:48:01 2016 -0500
```

```
    Fixed string ops test

commit c79ca4e7a8dea2ec71020052fec7862d4d62c946
Merge: 885dc57 46ae8fa
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sat Dec 17 22:38:12 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 885dc578a3b0907c3fa221854ad817438d84c091
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sat Dec 17 22:38:03 2016 -0500

    Fixed some tests

commit 46ae8fa3fd135f0a1e4d3f48d12d0ae06a95de28
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sat Dec 17 22:37:55 2016 -0500

    fixed some tensor tests

commit 710791d65a24af2a9b5679fd649191e2d50f8f50
Merge: def39ec 151358b
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sat Dec 17 22:03:19 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit def39ec037abd7025ed840a798842e5a26fee93e
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sat Dec 17 22:03:16 2016 -0500

    added fopen etc to semant

commit 28ab71990d39b1207a4a274a8929e0ffb949272b
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sat Dec 17 22:02:08 2016 -0500

    added tensor addition test

commit 151358bbb4bece828112ecd6696462fc2f837452
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sat Dec 17 22:02:02 2016 -0500

    Added a test for the order of function arg evaluation

commit 50add503662245022767d0e1ffd845386df32855
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sat Dec 17 21:11:15 2016 -0500

    logical operators all now return type int in semant. oops.
```

```
commit 0e8c9955babd4b03ff97517473d458e4e5cd131a
Merge: 50b4546 25c5468
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sat Dec 17 20:32:05 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 50b4546c7b33b4ee4e4e1fabbb75497b3d4bfaa4
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sat Dec 17 20:31:56 2016 -0500

    ints in tensors autocast to floats. i can write my bad tests again :)

commit 25c546827501cb303be78f6b59f36d3e6aa5a6e9
Merge: 937f23a 5ed87e5
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sat Dec 17 20:30:34 2016 -0500

    Merge branch 'master' of github.com:dyschwartz/LaTenS

commit 937f23aa1a4c542b5b46dcb28cfe4ccebc95c771
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sat Dec 17 20:30:28 2016 -0500

    Binary Ops with tensors working and tested.

commit 5ed87e5e54851c86434b7c2f8f2799e7399bf500
Merge: 8b02d59 8a567a7
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sat Dec 17 18:56:28 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 8b02d59924e73f56588ff2aa326deab6afb36956
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sat Dec 17 18:56:25 2016 -0500

    string to float/int and visa versa

commit 8a567a738c9bd20bfd8167ba9175761f50ed761f
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sat Dec 17 18:37:57 2016 -0500

    exit status now non-zero if failure is caught. fixes testing script.

commit c6970126f245d267b23fc671cb3a158bed1fca0d
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sat Dec 17 18:07:08 2016 -0500

    Removed sast warning for Eliana :)
```

```
commit a1c34819660a0172255e2c1d26ba5496a6c95b11
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sat Dec 17 18:03:51 2016 -0500

    Added itof and ftoi functions

commit 8944b3aa10f538b54b4f9efc004347d3cb67b477
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sat Dec 17 03:42:37 2016 -0500

    minor fix

commit a5c651efb4718b9db108a797289fa06373350e60
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sat Dec 17 03:32:20 2016 -0500

    Variable length tensor allocation working

commit 1caa38de5ae982f266f246f0d83c3704c0254a76
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sat Dec 17 02:16:51 2016 -0500

    Finally it works

commit dc8eda03e8cf07c3393132a7cd37da47870700f9
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sat Dec 17 00:05:23 2016 -0500

    Stupid llvm segfault

commit 99fefaf51dcd94a438c72eb8c8e61fdfbf576f42
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 16 23:17:41 2016 -0500

    More work, currently segfaulting LLVM yay

commit 4469d4d4455457eca615a9669ba4a5d9c443ce2d
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 16 21:35:56 2016 -0500

    Friendship ended with Structured Code, now Spaghetti Code is my best friend.

commit fffe6a11c0b6cf1aa5165c29fd24a25c6ca59bb0
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 16 21:28:25 2016 -0500

    missed these

commit a53a83aa8c4ca95d8f4217fdbffb1393833e1e34
Author: Mohit Rajpal <mr3522@columbia.edu>
```

```
Date:   Fri Dec 16 20:46:05 2016 -0500

    builder madness

commit 4c4a55c0fab9c919a74e7885030f7edd0da605fc
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 16 19:39:42 2016 -0500

    tensor access bug fixes

commit 40dd7309324f8b6472396f90912759fa19e79139
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 16 19:16:34 2016 -0500

    fixed and tested copy header of slice notation

commit 84bc39227d0673e29585414d15f2f83554133095
Merge: 0f5c3ab 4cfa3b4
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Fri Dec 16 18:41:15 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 0f5c3ab66a63e5799e66d72e323309e2a4ce4cce
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Fri Dec 16 18:40:09 2016 -0500

    dim() function in semant and codegen

commit 4cfa3b4477c7716fa90de9cdadb2228dfa61a128
Merge: ad76538 54772e9
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 16 17:29:55 2016 -0500

    Merge branch 'master' of github.com:dyschwartz/LaTenS

commit ad765387eb0620e7d4b01331ed383e5465d5808d
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 16 17:29:48 2016 -0500

    More codegen slice not fixes

commit 54772e932ead7b64c518229e1ba5e00aa4210a3d
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Fri Dec 16 16:44:49 2016 -0500

    fixes

commit ae46a4252947ceeabb54e5247fbd3758fddde230
Merge: 1fd0cc7 e269c69
Author: ElianaRose <erw2138@barnard.edu>
```

```
Date:   Fri Dec 16 16:41:00 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 1fd0cc72b1fe59dfc627ddbf68cc25b4ca7378bd
Author: ElianaRose <erw2138@barnard.edu>
Date:   Fri Dec 16 16:40:56 2016 -0500

    not only takes ints

commit e269c690a1db29184474219f2fd9ce036741c705
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Fri Dec 16 16:40:35 2016 -0500

    Integer logical operators

commit c0d71ed38158efcc541ced962b9ab8e42b1d6ef8
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 16 16:12:40 2016 -0500

    Compiling code for codegenning slice notation, testing/debugging now

commit 5e89a08b5cf1e32d690f897854581f8ada9d6b17
Merge: 5301062 82eecd4
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 16 16:05:22 2016 -0500

    Merge branch 'master' of github.com:dyschwartz/LaTenS

commit 5301062262c9c15ffec51d290fa8fed1c795a349
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 16 16:05:13 2016 -0500

    even closer to slice notation

commit 82eecd43ca5ff459579fbeefd9b9567bf29a418d
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Fri Dec 16 15:59:54 2016 -0500

    String concatenation

commit 4aa569a5659a76390e5cff1d51e789fea29e1903
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 16 14:08:56 2016 -0500

    More slice notation work ffs

commit 10a05f8af2c06ef04e112c4b3416ffe532b218a1
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Dec 15 22:56:57 2016 -0500
```

```
    Branching codegen_work to do changes

commit dff1fbe1124fd6021dfe942ad9d6c7aef2820e5a
Author: ElianaRose <erw2138@barnard.edu>
Date:   Thu Dec 15 21:41:50 2016 -0500

    removed semant warning

commit 82a4f6655fd7b83c9346b0e39f8aa5a437dcc546
Merge: fac1eb2 b2d1b1b
Author: ElianaRose <erw2138@barnard.edu>
Date:   Thu Dec 15 21:40:16 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit fac1eb2ad79beed75610dbe0fab32b7e91980c43
Author: ElianaRose <erw2138@barnard.edu>
Date:   Thu Dec 15 21:40:08 2016 -0500

    1/2-tensors now print as vector/matrix

commit b2d1b1b052bbcc4a31631b9155397dabf4cc1ec0
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Thu Dec 15 20:34:55 2016 -0500

    Improving the test suite; nor sure why error messages aren't working

commit 565332987572b86e5b78a190f74c8ca5cc79fc03
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Thu Dec 15 20:06:05 2016 -0500

    Added some new tensor tests

commit 74efa81b90ab7337f0aee0f36a59636c4448d238
Merge: 4be3f38 ef61347
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Dec 15 19:50:07 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 4be3f38e2de8ba5a3017c020ccf12c7ea3b5fa3c
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Dec 15 19:49:52 2016 -0500

    Added callback sast constructor, for codegen internal use

commit b2c7f158d851ee40f284b31aa405d936aed1b23f
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Dec 15 19:36:56 2016 -0500

    Tensor access and tensor storage working and tested
```

```
commit ef61347270f7f8036a3fee95b98598a5c9968a18
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Thu Dec 15 19:18:53 2016 -0500

    More exception fixes

commit d6e2c2d7135ced6ea7b95a40d4376c2f83710238
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Dec 15 19:16:54 2016 -0500

    pointer deref bug fixed

commit 98f5d99b272807ad3323203bf1badec9141bd3b0
Merge: 05f7fd3 5e0890e
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Dec 15 18:53:01 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 05f7fd3a553d7b6d8ff796bd581fb7d36a2a5bc8
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Dec 15 18:52:39 2016 -0500

    wtf?

commit 5e0890e91d64d44c1fa90e88eda083d280603605
Merge: d8797dd 70853d6
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Thu Dec 15 17:25:09 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit d8797ddcc3d751d5c9704482a99f98366806130e
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Thu Dec 15 17:24:56 2016 -0500

    Stopped calling List.rev everywhere and finally only called it once in the parser

commit 70853d67a65e3a0c25f97850bc708c82b8c9eaef
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Dec 15 17:09:48 2016 -0500

    Should be able to assign to a fully qualified tensor location now

commit 5f85f2c943478b2a7e4d78faf6659ff89ce37e90
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Thu Dec 15 16:44:32 2016 -0500

    Added line numbers into sast. Working. Line numbers not fully added.
```

```
commit 51c267111c2608f657db925d895136d963fd7950
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Thu Dec 15 15:55:53 2016 -0500

    Replaced last semant Failures with Exceptions, added a few line numbers

commit 2673d2d3a9d73381ded200774cf34510a8108bbb
Author: ElianaRose <erw2138@barnard.edu>
Date:   Thu Dec 15 15:17:49 2016 -0500

    changed line spacing for pretty printer

commit b12798d38b2981fba5c16923ca7cf30f713b2ab1
Author: ElianaRose <erw2138@barnard.edu>
Date:   Wed Dec 14 23:46:27 2016 -0500

    fixed return type checking

commit cb8b5a53ad5de99ea5dd23213fb15366a9697150
Merge: b45f79a 44c4027
Author: ElianaRose <erw2138@barnard.edu>
Date:   Wed Dec 14 22:17:25 2016 -0500

    merged

commit b45f79a38f878d2fec0b78d7faaa7c44c954e98d
Author: ElianaRose <erw2138@barnard.edu>
Date:   Wed Dec 14 21:52:28 2016 -0500

    added colors to pretty printer

commit 44c4027e79125943497297eb4c4533349aade546
Merge: 8d275b7 2759b92
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Wed Dec 14 19:56:47 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 8d275b7177024004d4442ab2c6d603d3fea6b3ad
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Wed Dec 14 19:56:39 2016 -0500

    Added index checking to global tensors

commit 2759b9271ea38c5ce03f06964d281f9fe096958f
Merge: f8ac66f 6134150
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Wed Dec 14 19:54:31 2016 -0500

    fixed merge conflicts
```

```
commit f8ac66f15c5c78094a4b7cd5b99429b043d989b7
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Wed Dec 14 19:52:38 2016 -0500

    line nums added to expr and bind. still missing from stmt, fdecls

commit 6134150a6b7945054eb9e688399cb6619f5e41e4
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Wed Dec 14 19:06:58 2016 -0500

    Removed last semant warning for Eliana

commit 06fb35707bf92737682fe0d739da0753073471d2
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Wed Dec 14 18:29:42 2016 -0500

    No more semant compile warnings

commit 513624838c0f035730e368bf0b31b5fd7c9f505f
Merge: 82cd30c 97005bb
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Wed Dec 14 18:14:26 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 82cd30c4a44bbf1e459c88fb433bbe0e5869fcfe
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Wed Dec 14 18:14:24 2016 -0500

    line nums in progress

commit 97005bb37c69b98bf2767cce9338eb5e36595958
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Wed Dec 14 18:02:00 2016 -0500

    Error message fixes

commit 00aa3deaaee9e4449de4530a4a0c2a6ab015f5b9
Merge: 66bc18d 65b3fe8
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Wed Dec 14 17:29:00 2016 -0500

    Merge fixes

commit 66bc18d4d2762c250d94590df056f8e83aeaee70
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Wed Dec 14 17:26:13 2016 -0500

    Put error messages back at the command line, fixed disambiguation error, added operator back into B

commit 65b3fe8b3e96b742b3e2f8cdf480bbdf47fa94ec
```

```
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Wed Dec 14 17:17:25 2016 -0500

    merge finished

commit 57253b0b0069328cb59c3ae205d26b4df8d5d21b
Author: ElianaRose <erw2138@barnard.edu>
Date:   Wed Dec 14 00:25:04 2016 -0500

    pretty print tensors

commit db7b39e3942709cffeb3e1daf450acfc895f1edd
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Mon Dec 12 23:44:14 2016 -0500

    Fixed merge breaks, compiling

commit 53f2f5e7f50ffee9dfd1938817db72262de2788e
Merge: 349e087 72e4c6d
Author: mr3522 <mr3522@columbia.edu>
Date:   Mon Dec 12 23:38:41 2016 -0500

    Merge pull request #4 from dyschwartz/fixTenSemant

    FixTenSemant - tensors implemented in semant

commit 72e4c6d8f1a95a977741718c74fe486d929fa85b
Merge: 29a7040 349e087
Author: mr3522 <mr3522@columbia.edu>
Date:   Mon Dec 12 23:37:34 2016 -0500

    Merge branch 'master' into fixTenSemant

commit 349e0870daa38955bf7519fcef5604c6b7753272
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Mon Dec 12 23:27:11 2016 -0500

    Tensor slicing headers copied, now actually need to copy contents

commit 29a704006dc0d72267019124c5bd2dca1ac71e13
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 12 23:22:57 2016 -0500

    Converted rank 0 tensors to floats; prettified type matching error messages

commit 35bf7185eb0d1045b1b2cebf430f6264d13e86bf
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 12 22:23:57 2016 -0500

    fixed parser
```

```
commit 9e5f291152623ec1dd6462c35218537adafaee4b
Merge: a739bfb 2704d47
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Mon Dec 12 22:07:56 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit a739bfbcfb55a5bbbd9a9adc7dd9e0aeafc566ba
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Mon Dec 12 22:07:43 2016 -0500

    Tensor slicing working

commit 2704d47edbcf40a051d2aefc3276d3429d30fa98
Merge: 196e198 04b6369
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 12 21:56:39 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 196e1982d17bb5c583af7075afe752689723acaa
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 12 21:56:36 2016 -0500

    scanner fixed to escape special characters in strings, .gitignore added, many tests updated to our n

commit 04b63696e5fcb354ad4d94fe645d6e70d4cad6d9
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Mon Dec 12 21:31:34 2016 -0500

    Tensor access working

commit 87c5d89964c798e9b2dd75958fbe3071e1a445ae
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 12 21:27:44 2016 -0500

    Fixed tensor binds for Mohit; tensor immediates doube-counting

commit 680726fccf3204e0282fccdfabf20fbfd3cf3d35
Merge: 71f7b0a 12ab316
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Mon Dec 12 20:53:26 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 71f7b0a650f5598e0ce5f63d108a9934ca2fa404
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Mon Dec 12 20:50:01 2016 -0500

    fixed function parameters not used
```

```
commit 12ab3165bb29e06c3c08a6a40853ea60ce97848d
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 12 19:56:38 2016 -0500

    boolean binops properly get typed as int

commit e703dc73b215e6d94b71b3f1d4ff9c3125084023
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 12 19:34:29 2016 -0500

    i mean now. now the last trace of microc is eliminated

commit c3eb542e60d07fd67dd8a0b8d6b57776dcbd46ec
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 12 19:33:10 2016 -0500

    print_string put back in semant, final microc vestige removed from testall

commit bb317e7855b06e6f313cdb521bf64c280f7c87e4
Merge: 6809598 4502b5a
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 12 19:31:25 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 4502b5a7dfb9d63fd453ffc9a87a5306d57dba34
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Mon Dec 12 19:30:30 2016 -0500

    Fixed strings

commit 6809598078ce2d9dde549ab511c64bcddf440442
Merge: 763e9d3 d945976
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 12 19:10:35 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit d945976fc853544347ef04f034b8dba02b505060
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Mon Dec 12 19:10:26 2016 -0500

    Fixed FPs and added new style tensors allocation

commit 91f232f68707f9b7445fcf9a0d4e9992fe426e91
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 12 18:56:33 2016 -0500

    breaks! fixing parser

commit 763e9d3c43366429623278a76ef98ff535e72b86
```

```
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 12 18:53:51 2016 -0500

    - scanner updated to handle escaped characters and floats with unsigned e
    - semicolon added to fix broken test

commit 4ee0342e8c0fe8f2efbfacd037e1801c7f46897c
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Mon Dec 12 18:40:57 2016 -0500

    codegen supports floats, strings TODO

commit 133bb7df8e95121bbcf989f9b35c9d4fe462aabb
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Mon Dec 12 18:23:25 2016 -0500

    Further codegen updates, more progress

commit 46047ede6c83bb24d290a1e51737453f68d46a8f
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 11 17:35:15 2016 -0500

    exceptions migration finished

commit 178af2f12f2325ac65491b1341b86e5ef27742b7
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 11 15:54:30 2016 -0500

    fixed float/tensor multiplication

commit 94adeb0bef802cde529aba192514f9b88b7b7671
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 11 15:48:11 2016 -0500

    tensor multiplication

commit 3af8f725b59940e5461d3f928aa4b2f3f6273ca3
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 11 15:24:18 2016 -0500

    Fixed the stupid??? list??? error??????

commit c11ce4e145212edc1464291a33fab01f1d6b6bba
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 11 15:07:36 2016 -0500

    tensor multiplication

commit df3f61acaaac7b9a0b4b438a26cd2296e4834236
Merge: c36293a 086ebd5
Author: Daniel Schwartz <ds3263@columbia.edu>
```

```
Date:   Sun Dec 11 13:55:38 2016 -0500

     merge conflicts fixed

commit c36293a1f37d92998537877b7045aa1bc409eff5
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 11 13:53:30 2016 -0500

    more exceptions migrated

commit 5c96f7c56f92b90679c2d31faa138f7519255f40
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 11 13:48:08 2016 -0500

    typify fixed

commit c3827ef3ed1c3409af3f2c5134c8976e678d7142
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 11 13:40:22 2016 -0500

    progress on exceptions

commit 086ebd5d22b6a30d46b646398156b6b7552f544c
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 11 13:14:26 2016 -0500

    iterators compiling, will test and implement tensor functionality

commit 90d3c4b0c4c8fd3adffb85d59c3c1c5105904b28
Merge: 2bb56be b1f8ab4
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 11 12:34:13 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 2bb56be02a0cbd0078a7700fa945877412ecd9c0
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 11 12:34:01 2016 -0500

    Changed test programs to conform to actual language

commit b1f8ab406428fbdb8c20c41eab11f50eb7e15d6a
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 11 12:33:52 2016 -0500

    for loops. yay

commit 6626a3396f418d02d2ddc479852ff66803c2a71e
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 11 12:16:25 2016 -0500
```

```
    took open ast out of semant

commit f158dee8cfd5cc8f37b10ece13b2c2fbcecb276a
Merge: 3ec8cb1 cc32cf4
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 11 12:11:15 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 3ec8cb193cbf28e06a5268391bd9947d9319c856
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 11 12:11:09 2016 -0500

    pretty printer fixes

commit cc32cf4caa89d40a4835346b74ea092e392ea086
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 11 11:52:29 2016 -0500

    4.01 compatibility

commit 58222b56717386bacbd57ad96ee5725b68e8011e
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 11 10:50:19 2016 -0500

    tensor iterator work in progress

commit 4f8ccd51a7afe23c307abe6b15ae98f6392f94f3
Author: mr3522 <mr3522@columbia.edu>
Date:   Sun Dec 11 10:20:39 2016 -0500

    Update sast.ml

commit 4db5a9db095709bf539b0a91f10761adedb41f49
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 11 07:37:32 2016 -0500

    Floats supported, type promotion working

commit 5ba40a0d02519e36fade4d59ac351133c8fd4fc0
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 11 07:06:11 2016 -0500

    Tensor storage to variables working

commit c8bb1c1d27996294417c4dd32f6911e53b8a3034
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 11 06:19:19 2016 -0500

    Added tensor lookup
```

```
commit 11ee0ed6b8127e66272683c9c964a0f0cc266679
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 11 05:42:37 2016 -0500

    Should build any immediate now

commit 7b7da7bfc3408cb624c012229f8841857aec28ef
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 11 05:01:55 2016 -0500

    Tensor immediates working

commit b955efc6924828ad0e865d4f9b5b0d6e7145b959
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Dec 11 02:49:11 2016 -0500

    fixed break by commenting out code, please stop breaking the master branch

commit 56c7570cf8c60e8ec5fe96c13da15d8495573bcd
Merge: 78cd9c6 3ab47d9
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Thu Dec 8 18:31:42 2016 -0500

     ? trying to figure out merge ?

commit 78cd9c6bb6033c797beba52ebab25c88238c7ea3
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Thu Dec 8 18:28:39 2016 -0500

    parser errors work except for printing actual bad token

commit 3ab47d9b795381e358ca285d787d6f876c7af53c
Merge: 2098460 d54a59e
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Thu Dec 8 17:37:23 2016 -0500

    Merged tensor with master branch

commit d54a59ef6acbaf099d2d8efb57e246d4d1e9efac
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Thu Dec 8 17:11:24 2016 -0500

    Added tensor checks to check_globals

commit fb742406db5e6ff48485397bace599d486cff27b
Author: ElianaRose <erw2138@barnard.edu>
Date:   Thu Dec 8 16:47:26 2016 -0500

    fixed some warnings

commit 209846042a0b6f46e30d60bd2ed8d5c57aeeef34
```

```
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Thu Dec 8 16:39:36 2016 -0500

    linenumbers from scanner + scanner cleaned + somewhat better exception handling

commit 21730bd3b708cc0380ce88c5d5849c63d4575721
Author: ElianaRose <erw2138@barnard.edu>
Date:   Thu Dec 8 16:26:35 2016 -0500

    tensors in semant workgit add semant.ml

commit 54f9298af729480a31e44dc9e5ee1c47ac6429fc
Author: ElianaRose <erw2138@barnard.edu>
Date:   Thu Dec 8 15:58:12 2016 -0500

    TensorAssign implemented in semant

commit 0c54fde2df1e942390e48a10ca6d920489c3f533
Author: ElianaRose <erw2138@barnard.edu>
Date:   Thu Dec 8 15:28:32 2016 -0500

    check that dimension/size of subtensors are equal

commit 887c41756131e2160153a72d69215e21f9b0dc2a
Author: ElianaRose <erw2138@barnard.edu>
Date:   Thu Dec 8 14:53:00 2016 -0500

    check that expr in tensors are ints/floats

commit 494f073fb91884009e040091c0024cac2406436e
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Tue Dec 6 09:53:18 2016 -0500

    fixed to only warn when statements follow return in block

commit 7490b7f52b04320b0f3e5a0d27ed371e569f7105
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Tue Dec 6 09:28:46 2016 -0500

    now errors if statements follow return in block

commit 9528a39166d7373d2c8946a7979f842c71a4001c
Author: ElianaRose <erw2138@barnard.edu>
Date:   Tue Dec 6 00:08:43 2016 -0500

    tensor immediate checks for rank

commit 2f3662daeaad4143955fb827a03b3786b549d3f1
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Tue Dec 6 00:04:59 2016 -0500
```

```
    latex comments made consistent for fdecls, README updated for eventual inclusion in LRM

commit 9fe4c5c942b2fe78261721a9170e48b1a1fc0e60
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 5 23:58:00 2016 -0500

    pretty printing comments merged into regular scanner/parser/ast

commit 573a9bfa6d13be8ecba6e3059751a5bcb3a4906e
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 5 23:38:24 2016 -0500

    comments before blocks and before func decls

commit e28f4166c2f603ab092d3943afa2aa849e981128
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 5 23:16:45 2016 -0500

    half of tensor immediates work

commit 95e0be653d1ccd7118dfdd340f93d410e96dac49
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 5 22:58:26 2016 -0500

    Started tensor immediates

commit 1007f900f4af444f02650885651a17d2b5a451fb
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 5 22:36:07 2016 -0500

    Fixed tensor rank calculation in TensorId

commit 24d79bcf057dc30cb0e0e97f7c09644a65c69825
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 5 22:18:36 2016 -0500

    Tensor IDs in semant, no tensor literals yet; changed tensor type in SAST to contain only int instea

commit 36901622bafb0f9f7ccd437ea4ba1603982d817a
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 5 22:00:34 2016 -0500

    README has link for llvm reference manual

commit 7c2fb1f23217f3fb67f60f4c54f5a2b762c1c7dd
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 5 21:59:46 2016 -0500

    print_int working. strings in general don't work in codegen

commit 92fd21142a2d0393661d9f717e3faa0bfedb2b46
```

```
Merge: 12620a8 e6f104e
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 5 20:21:43 2016 -0500

    Merge branch 'tensor' of https://github.com/dyschwartz/LaTenS into tensor

commit 12620a8ac17f1397418f1c8997beb3a56f34f772
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 5 20:21:25 2016 -0500

    started fixing sast for tensors

commit e6f104e713e9707463a5316ae5208e5297196ecb
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 5 20:21:04 2016 -0500

    comments

commit be6ab11d4502eb25cac4197d7c49ba2a70265717
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 5 20:11:40 2016 -0500

    Prevented user from redefining variables in same scope

commit 386e82829b84a7251c0acaf75ba372dc71be2ce5
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 5 20:06:17 2016 -0500

    finished printing ast with tensors

commit dca413f54b5c8d6001afdb81dda3473bcdc4e54b
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 5 19:47:01 2016 -0500

    fixed parser to handle tensors better, made ast print tensors

commit 809c506205a5c9ab10806cf82d876e4ac7ba6a63
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 5 18:36:53 2016 -0500

    removed extra line in past

commit 840266a6ebd2b0a5934ae509a9bfc4aeed95a301
Merge: a20bd06 62e616f
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Dec 5 18:33:34 2016 -0500

    merged

commit a20bd06e7e6148ef17252a525fcff43912eb37e4
Author: ElianaRose <erw2138@barnard.edu>
```

```
Date:   Mon Dec 5 18:30:01 2016 -0500

    fixed pretty printer to actually print

commit 62e616fb1e56c0793d771cfe44ac15931d51ca02
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Dec 5 18:10:49 2016 -0500

    printf now takes a single string as its argument

commit f3a3c0310f3c880a12fd241ef89472c861e0d210
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 5 14:52:39 2016 -0500

    Added debug global environment print option

commit 8b7fffab6170d822a46903ff0abaa10700a56169
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Dec 5 13:08:52 2016 -0500

    Test suite works. By which I mean everything fails, but the script runs. Edit the llvm depending on

commit a47ca083466a2e3539fa12132f3ec2e5d5ea85da
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Dec 4 21:05:09 2016 -0500

    pretty printer with no comments or tensors

commit 8a882801d5e5b3eafe66e527d1b19070e5e80136
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 4 14:28:12 2016 -0500

    check_call cleaned

commit 46d36511e46d3304696660b4fc20cbb80fff6235
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 4 14:24:31 2016 -0500

    printf working in semant

commit e0884620ade1cfdcc1a9b3a2ebd60162b2cd5f25
Merge: a5a2cce 364345b
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 4 13:39:31 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit a5a2cce07606be3e9e45660b755d15d643bad5f8
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 4 13:39:13 2016 -0500
```

```
    Threw away the quotes from a string literal

commit 364345b19f14d323f3fd56d3fb59b2301fae3b57
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 4 13:24:22 2016 -0500

    printf passes semantic check, pipeline for adding built in functions

commit 1efab0133c6ee5721c21f708ae636985bfbd98b2
Merge: cc2bbbe ed34388
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 4 12:40:30 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit cc2bbbe53999904ec49816fe68339873a0998055
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Dec 4 12:40:25 2016 -0500

    semant handles overloaded functions

commit ed34388c91e4c79f0b98bc657f8ef2b25952ee27
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Dec 4 11:57:40 2016 -0500

    Fixed sast pretty printer now that binds are sexprs

commit 897baf4e95a0973e58928db98a3d0b994a48ee74
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 2 14:04:36 2016 -0500

    Added support for print special function, currently can't work because semant pipeline does not lik

commit 6c3016e7799e15f4f02ff06d176379ac7b776b39
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 2 02:19:40 2016 -0500

    functions and function calls working, calling it a night

commit 55e686f982265655dbaa8547ff626c67569cc3de
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 2 02:10:54 2016 -0500

    Tested For, While, If-then-else, Return, all the basic expressions, blocks, Binds, bindsandassigns,

commit f8caf05d88caf3ffe1d524ffed47f33c63c245ac
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 2 01:19:05 2016 -0500

    Modified binds/lookup to use new stack for all operations, is compiling, will test
```

```
commit b7da6f0c4355f4dc498228e0bb5b6c6667a465fa
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 2 01:04:01 2016 -0500

    Added mutable stack as holder for scope, we need it as scope is changed dynamically by expressions

commit 4184bcd943f4d465908278f6350c27031e4dc9a2
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Fri Dec 2 00:34:57 2016 -0500

    Fixed another bug with semant throwing away an sexpr and keeping expr, we should uniformly keep arou

commit 1a355493e8325c8aa5b8181a799d58fb40d54163
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Dec 1 22:49:11 2016 -0500

    Literals now working inside functions, tested verified

commit 106699b934f83bda50b882f51c1b0a15d47aeab4
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Dec 1 21:55:48 2016 -0500

    Scoping lookup compiling, probably working

commit 68c2347ab760e1548abbfe327444ff7c9a57d034
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Dec 1 21:33:54 2016 -0500

    globals tested and working, fixed floating point parsing bug in scanner

commit 7fd3ef01248fc869e89177cddb5698d5d8c9e394
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Dec 1 21:13:02 2016 -0500

    globals added and compiling, testing now

commit 22f6da97fda716dd39b2aafca0a37dca1b5cae83
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Dec 1 20:10:45 2016 -0500

    HUGE fix to semant/sast, semant/sast completely threw away the expression associated with any varial

commit 61b9d358c4734da64ab6bbbb6171501e06136e69
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Dec 1 18:57:24 2016 -0500

    Fixed codegen to something that works

commit 7823572311e3727171796ef56477fdbc049566dc
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Thu Dec 1 17:09:46 2016 -0500
```

```
    sast printing works. booya

commit 8746136a4f45d69532f637541442133d069ccd37
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Thu Dec 1 16:15:53 2016 -0500

    Pretty compiler warnings, prettier semant.ml

commit e8170e9ea8d80918061407af274b09a0d8d31463
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Thu Dec 1 16:15:17 2016 -0500

    README updated with instructions to pretty print

commit 5fcbfddac4c773819c0a9f2d7ea916717e2fa8c9
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Thu Dec 1 15:51:23 2016 -0500

    fixed sast pretty printer and added latens flag to run it

commit 19e2a3de29710acf3ad5ea7a6818e498640b1ad8
Merge: 39620eb 5176210
Author: ElianaRose <erw2138@barnard.edu>
Date:   Thu Dec 1 15:39:42 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 39620eb8dbd6e0e5824d3f0737eb7ba1ab35868d
Merge: 7af9cc6 5b6e47c
Author: ElianaRose <erw2138@barnard.edu>
Date:   Thu Dec 1 15:36:45 2016 -0500

    Merge branch 'fixSemant' of https://github.com/dyschwartz/LaTenS into fixSemant

commit 7af9cc67875d6448b15609ac60618f6b1407bcd2
Author: ElianaRose <erw2138@barnard.edu>
Date:   Thu Dec 1 15:34:45 2016 -0500

    sast pretty printer

commit 5176210101ee85c0905ccf14eafa0547a00820d5
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Thu Dec 1 15:28:11 2016 -0500

    latens fixed to handle printing sast

commit ac112e45df4db2c80d639beaa07d0dbec2413d3a
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Thu Dec 1 15:20:49 2016 -0500
```

```
    added formals names to Sast.call

commit 5b6e47c6ed3e1294a468a73f1269444d1666cc18
Merge: ef281f0 3d22980
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Nov 28 22:34:09 2016 -0500

    Merge branch 'fixSemant' of https://github.com/dyschwartz/LaTenS into fixSemant

commit ef281f039002b25858258b7ebdc9adc3d6c83f61
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Nov 28 22:33:49 2016 -0500

    Parser understands declarations

commit 3d2298068ff4a5a0f5da602183161bc9838a3faa
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 28 22:13:47 2016 -0500

    fixed warnings

commit 4f4bcfa322c05257eca151dd8067cf3a0759147b
Merge: 52f1110 ae05fd3
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 28 21:39:39 2016 -0500

    merged with master

commit ae05fd3184cf93b9e44e71bbc74ec3c68b35b5dd
Merge: 62d04de af6a734
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 28 21:34:45 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit af6a7348626de14daa90788b4b927a2df055a2ea
Merge: acdf0bf 7d09bb3
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Nov 28 21:17:33 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 52f1110bbfb4a91e252d68fd9c752472e80e344d
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Nov 28 21:17:04 2016 -0500

    Fixed parser to compile with function body being a stmt not a block; no codegen

commit 3681450ac3d053ccf491bbb1eb170f101d34567b
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 28 21:01:19 2016 -0500
```

```
    started fixing semant

commit acdf0bf6a6c7252a8333e3858da25cf5dbc83b73
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Nov 28 21:00:12 2016 -0500

    check_return checks return type

commit 13484b521969d9f62503bcfc58f28beb77a28cf2
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Nov 28 20:59:10 2016 -0500

    check_stmt no longer takes options for local_env, added function return type to local_env

commit 3a2c87a15819da7e2a79817d3481126d60bd34d0
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Nov 28 20:57:16 2016 -0500

    check_stmt no longer takes options for local_env, added function return type to local_env

commit 7d09bb35053fb49031d498272d104c47c12db476
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Nov 28 20:52:41 2016 -0500

    Added pretty printer functions to ast and sast

commit a99d85cf0a1f91359c645baf5557821a46f9d9cc
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 28 20:33:24 2016 -0500

    changed name of microc to latens

commit 62d04de450ae2b5dbb4aef155e3f2159d7fb9f30
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 28 20:32:30 2016 -0500

    working on making

commit 063f2b637bf50365b23450e8b42c06c6c1f32a74
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Nov 28 20:20:26 2016 -0500

    deleted redundant notes files

commit b3f328e467edb23cb25d1c70191578e128b70b1b
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 28 18:02:59 2016 -0500

    added notes file
```

```
commit 901ce30027eff7ba79e9d27dcc469c9c4fd1ffdb
Author: ElianaRose <erw2138@barnard.edu>
Date:   Tue Nov 22 00:12:21 2016 -0500

    apparent fix needed for semant

commit 1a94449007cf61ec59a12fba9d982f0d0e7bf933
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Nov 21 23:59:07 2016 -0500

    Made semant return a sprogram

commit b116b186d927cb12efb2e5caadc38fc20d727851
Merge: 36377c2 112446f
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 21 23:41:10 2016 -0500

    Merge branch 'fixSemant'

commit 112446f45755ac30fcf2142f882a1ebe6c6af5a9
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 21 23:37:13 2016 -0500

    merged

commit a5f48f643d73a00a0af97017568086e1bc806cc6
Merge: d116e0a 9e5bd9e
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 21 23:35:23 2016 -0500

    Merge branch 'fixSemant' of https://github.com/dyschwartz/LaTenS into fixSemant

commit d116e0a04839f9bfa39ed0c6aafdd3757a78b67d
Merge: c109509 caca94d
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 21 23:34:27 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS into fixSemant

commit c109509215a9d32c2bc0e8a31151d43165cc1f3d
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 21 23:33:58 2016 -0500

    it works(with globals)

commit 9e5bd9e9ed5caa3444c1a54dce75ffb30705a48e
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Nov 21 23:05:02 2016 -0500

    scanner comments addressed
```

```
commit 14911fa4d08de64fc6b338fb621d959453574766
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Nov 21 22:49:58 2016 -0500

    sast neatified

commit 23402a403474bb506e93bdf7875b83aced81f67d
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Nov 21 22:39:14 2016 -0500

    env is now global_env in check_expr, for readability

commit 2887fd34970dd18b9a677412300b643eee1e4dea
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Mon Nov 21 22:35:13 2016 -0500

    hierarchical searching to resolve var names

commit 18b1dd59b5c32443c880864b7d9ab768976380a4
Merge: 0da1317 50cc361
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 21 22:27:58 2016 -0500

    Merge branch 'fixSemant' of https://github.com/dyschwartz/LaTenS into fixSemant

commit 0da131776712a7f0e6366dc74f9f2fbaccda2623
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 21 22:27:45 2016 -0500

    added env hierarchy search when resolving var names

commit 50cc361d3130945de6a04f514598a6af731a345e
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Nov 21 22:15:11 2016 -0500

    Finished check_function, no errors

commit 072114e70b4f8abe63de41865b6b182703e4b10a
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Nov 21 22:03:30 2016 -0500

    99% written check_functions

commit dd5f6a8db118f4aa0b428d1a5437509d6b8ed79c
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Nov 21 21:41:13 2016 -0500

    half-working check_functions

commit 887359018b97bb10185b6f60ba8ce212810ef5ce
Author: Daniel Schwartz <ds3263@columbia.edu>
```

```
Date:   Mon Nov 21 20:39:55 2016 -0500

    semant compiles warning-free up to check_globals/check_functions

commit d4aa82693a64d25c096fff0e5563e80e83bf476e
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Nov 21 20:26:56 2016 -0500

    Semant compiling without errors again

commit 36377c2740220a7ba9ab96b46f7a3b28ead5dd45
Merge: 791b99d caca94d
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 21 18:37:55 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 685e5340063c13f68d6b82e21a6b163f267cf016
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Nov 21 18:37:38 2016 -0500

    Modified SAST

commit caca94dc98ce6b4fe39af2cb39a9194a3e964b8c
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Mon Nov 21 18:35:50 2016 -0500

    Hello World working

commit 791b99ddbd4689b9cbc182d2ace9eeb14f80160f
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 21 18:07:09 2016 -0500

    trying to fix

commit fe40bba093cbf518550633c06d250ead421d7557
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Mon Nov 21 16:42:19 2016 -0500

    shit compiles.

commit ad1b7d55b94b29596ae5fe78d536871039cd4e70
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 21 16:04:49 2016 -0500

    more fixes

commit 5e5e8cbae879d0439ae6bbf1c76df42678728a36
Merge: ba72dff 7c9325b
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 21 15:41:34 2016 -0500
```

```
    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit ba72dff2e6cff98795cd24046c651e89704c45a9
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 21 15:41:28 2016 -0500

    more fixes

commit 7c9325bb1025127e53e77eaa84a27015f5a5b6f4
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Nov 21 14:38:39 2016 -0500

    Beginning check_functions

commit f8da2ed673021be055723d0f9f511c8b4fc90069
Merge: b79c483 1b27502
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Nov 21 13:08:36 2016 -0500

    Merge fixes; no errors (no check_functions)

commit b79c4835eaac1c10be34a2d8fd375bbe7c736202
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Mon Nov 21 13:02:53 2016 -0500

    Compiling with warnings (no check_function)

commit 1b275027fd398263a8366dc3d8993b0a6951bf8a
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 21 12:56:32 2016 -0500

    more fixes

commit 386a0526bec0bd97d6fedcf656158c6ef1c9d812
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 21 00:25:01 2016 -0500

    fixes

commit 32efd9d5d2f701e2e65f75ea5bd90e7204601290
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Nov 21 00:09:50 2016 -0500

    fixes

commit 771ea2889dafbb26f2915657ab7dd004d429a97b
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Nov 20 23:59:01 2016 -0500

    fix
```

```
commit 6bca5e263a61342e9d5d243ee7371bb799914f6e
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Nov 20 23:52:05 2016 -0500

    More of the check_expr error

commit f194f7a724c63a4044c5fbe1bccafc98c733ada9
Merge: 48a5d1d 1a7e81d
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Nov 20 23:40:35 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 48a5d1d88fb970a6b67e810d6dd899f5751686ab
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Nov 20 23:40:31 2016 -0500

    fixed errors

commit 1a7e81d59921857378adc5be9de912de78cbc84b
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Nov 20 23:39:15 2016 -0500

    Fixed the elusive id error

commit 647d903e726ac5a29915e7309518d1ff8e943030
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Nov 20 23:29:44 2016 -0500

    Fixed typify: added something (sast) to return at the end, removed nonsense

commit 39b13ae5c85233cc355634ec7a05812bb031e57d
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Nov 20 23:05:18 2016 -0500

    fixed errors

commit cd9cb6ffa8d2481a6fc79d5643e6cc9e41e52bbe
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Nov 20 22:43:58 2016 -0500

    Added global environment to keep track of function decls + their args

commit 7a5d510e3afa7e6828ee6c15a41da06cb10b2436
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Nov 20 22:18:13 2016 -0500

    fixes

commit ef57d014626b13768b73026b08c071a2c8325fd8
```

```
Merge: bf1c38e 2cb2cd7
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Nov 20 21:23:20 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit bf1c38e108aee11ddf953c8c144b7fac77af5fc6
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Nov 20 21:23:04 2016 -0500

    still trying to fix semant

commit 2cb2cd72ac7da4ddc98f47831da4d8883cd420b8
Merge: cfc9775 47ed183
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Nov 20 20:53:41 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit cfc977504976ed5e52d80871eae7cc67447bf66a
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Nov 20 20:53:18 2016 -0500

    Hardcoded the structure of a hello world sast into microc.ml, will codegen this

commit 47ed1836514f9a1aca4d713d111a8ddabcd41be9
Merge: 42ef333 89035d1
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Nov 20 20:48:23 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 42ef33326a9dcca3e5cced836a5ab77b811f3fd7
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Nov 20 20:48:15 2016 -0500

    fixed line 177

commit 89035d1ebb37d3a411d4815adcc6445848b435c4
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Sun Nov 20 20:33:06 2016 -0500

    codegen/microc compiling

commit 054a0a19304e2515306491eeff3b23659dec60f4
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Nov 20 18:41:28 2016 -0500

    small fixes

commit e3b7ef05ae34adbbe0f956d57cdf4c33af8dedbc
```

```
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Nov 20 18:10:55 2016 -0500

    little fixes

commit daa642082cb064024c7abce14f2650d86cc9061b
Merge: 2a2d859 17510c6
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Nov 20 17:56:58 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 2a2d859e62e54396992af7f4a0ebe25ef38b8e0c
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Nov 20 17:56:50 2016 -0500

    added up to Call in Ast.expr

commit 17510c66b318fbc6d478b58192f619216bb64e8b
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sun Nov 20 17:56:41 2016 -0500

    Added check_stmt

commit d37cd55e616aaf01537182c2076370f99ae616b2
Merge: 247f9eb 87206b6
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Nov 20 16:33:28 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 247f9eba50b987c9f8817dce26020d48c2e46436
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Nov 20 16:33:16 2016 -0500

    started check_expr

commit 87206b607d70a3e96ab4cf88bcd75e0b019c239c
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Nov 20 16:30:30 2016 -0500

    updated changelog

commit 14f367841a33b0c6b9a6326314bd6ef5518d9ff3
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Sun Nov 20 16:28:35 2016 -0500

    check globals started

commit 53110cbde208f39872a93b5b83aec5bacd6252fd
Author: ElianaRose <erw2138@barnard.edu>
```

```
Date:   Sun Nov 20 13:01:28 2016 -0500

    defined check_stmt and check_sblock

commit 165fd6fa948430ee77169adc0d1864ad2cd91f24
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Fri Nov 18 11:35:44 2016 -0500

    Added scope to the SAST variable declaration to help codegen resolve variables

commit 964de87a383692a0471bba806d0bec40c5554996
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Thu Nov 17 18:15:15 2016 -0500

    Friendship ended with MICROC SEMANT, now OUR NEW SEMANT FILE is my best friend (see changelog/seman

commit e29c54db284a9684ee640e9794e05676751b9490
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Thu Nov 17 15:29:53 2016 -0500

    fixes

commit dc95c6770ed03f8a57d77207ecd12bb3abf4c9de
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Thu Nov 17 15:28:56 2016 -0500

    More merge fixes...oops

commit 058fd9d27992eb68864208d05585972d58b565b6
Merge: 6a29fca b350040
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Thu Nov 17 15:24:35 2016 -0500

    Merge fixes

commit 6a29fca63c4b25ae9e7574062ae3bb3963395e6d
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Thu Nov 17 15:21:35 2016 -0500

    Updated sast with expr * type

commit b3500403fae0aa2f907a6e641333eaddff20e1b0
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Nov 17 12:54:40 2016 -0500

    progress on codegen, (NOT BUILDING)

commit 256d3d97a98dd09f4ba1f448c9d259680f36451c
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Nov 17 11:52:49 2016 -0500
```

```
    fixed break

commit fc0d9828b9b3ae07e8b80cc857263c746da14dd1
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Tue Nov 15 16:36:52 2016 -0500

    Compiling (and very empty) sast

commit 6854769885da65f2194c4f693241117cef4fac2e
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Tue Nov 15 15:16:39 2016 -0500

    Test suite V1

commit dd35d0687b0ee6ea16342d5be754b8728dbe8b7f
Author: ElianaRose <erw2138@barnard.edu>
Date:   Thu Nov 10 17:22:07 2016 -0500

    fixed sast

commit af96cd1e168c81c025cd669cf4822f9c05cd31cb
Author: ElianaRose <erw2138@barnard.edu>
Date:   Thu Nov 10 16:55:55 2016 -0500

    added sast

commit 24bcd618a3893cd4ba7aa0a6587e1dc03b4e2364
Merge: 60e8927 c31f7c0
Author: ElianaRose <erw2138@barnard.edu>
Date:   Thu Nov 10 15:57:04 2016 -0500

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 60e8927538930089af26f87a1e2ced4809f9fcbe
Author: ElianaRose <erw2138@barnard.edu>
Date:   Thu Nov 10 15:56:53 2016 -0500

    added back the literals without expressions to tensor indices

commit c31f7c0663dfb67804dbff467c2d74b7a49f1e3a
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Thu Nov 10 15:38:00 2016 -0500

    README updated

commit c01da4244c0e3994116af46d9d655e347d04b6c6
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Thu Nov 10 15:37:15 2016 -0500

    README updated
```

```
commit 017d50e0771d36e408bc76d16684bb3083d57ea2
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Fri Nov 4 00:19:43 2016 -0400

    README updated w/ important links

commit eefff1dca67201d59cc798993fc62a466eea833c
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Nov 3 16:40:20 2016 -0400

    fixed parser

commit 3625071a0585a06b551cc375503400ebec5d7349
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Nov 3 16:32:14 2016 -0400

    made some fixes to parser. still broken

commit 1553107679e149fbe697ce1468ef32bc3473e2be
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Nov 3 16:17:34 2016 -0400

    fixed

commit feb50a373be2244ed82d2977b44a224d8773d3a0
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Nov 3 16:16:02 2016 -0400

    compiling up to parser.mly

commit 8447bb5d4e26a4f06045dc9e65c5f16e656ba4e9
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Nov 3 15:54:26 2016 -0400

    fixed scanner

commit a8b7a929d87b728ef14ce76b30bd53b3a65f2a16
Merge: f6f788a 335c7db
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Nov 3 15:50:48 2016 -0400

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit f6f788a88c01225e77dd040b36c211ff6247da22
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Nov 3 15:50:33 2016 -0400

    codegen compiles

commit 335c7dbfca11052e40314a05889d3d682488a2c3
Author: Daniel Schwartz <ds3263@columbia.edu>
```

```
Date:   Thu Nov 3 15:48:26 2016 -0400

    semant.ml compiles. gutted functionality with commenting so it can be fixed+restored piece by piece

commit a1c69222c8e363188b6a1875672e1511a548450f
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Nov 3 15:19:53 2016 -0400

    codegen update

commit cbffa539dcd99591cc34987755bc46de1722b101
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Nov 3 15:07:00 2016 -0400

    codegen update

commit 5c655cd57b49140f0a00ccbda43449b4082e3faa
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Nov 3 14:57:41 2016 -0400

    fixed ast.ml made formas from bind list to (type * string) list

commit 4fc2864d849f6382cacb7dff2c3e10ae874ec839
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Nov 3 14:49:44 2016 -0400

    fixed var_typ

commit 5ef40e29e6cb18ae46d53ba211b0d4698c2016b9
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Wed Nov 2 14:50:01 2016 -0400

    changelog has debriefing for Mohit

commit d6202d7785e1f132235283e7c0f6f5bff126440a
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Wed Nov 2 10:31:26 2016 -0400

    changelog updated

commit 1ec2029ee785d456a1650936decf1ae3255d25ea
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Tue Nov 1 22:40:17 2016 -0400

    TODO moved to changelog

commit 7eb140fbe9c4b713a12dcc79572460ea56d5576d
Author: ElianaRose <erw2138@barnard.edu>
Date:   Tue Nov 1 21:56:08 2016 -0400

    added stuff to changelog
```

```
commit a0885ada1ae7b7e2e1bff60a2e41348d5bfa5cc2
Author: ElianaRose <erw2138@barnard.edu>
Date:   Tue Nov 1 21:14:24 2016 -0400

    added int to ast, removed bool from codegen

commit 64f734813e23b61fcf628b9115eb1dba4d149197
Author: ElianaRose <erw2138@barnard.edu>
Date:   Tue Nov 1 15:35:04 2016 -0400

    removed precedence rule for UNDER

commit 65d2482c863c007375cdd3839979c8eac1b6fb51
Author: ElianaRose <erw2138@barnard.edu>
Date:   Tue Nov 1 15:29:52 2016 -0400

    added int to parser

commit 10f109411dcd945049d9fb676bece8de86c77739
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Tue Nov 1 15:18:19 2016 -0400

    Sedward's makefile, added TODO list, added INT to scanner

commit ee3ba33ba80a4eeeee2793f91d4f55aec8fe47b6
Merge: 542edd3 dacfdcc
Author: ElianaRose <erw2138@barnard.edu>
Date:   Tue Nov 1 14:46:43 2016 -0400

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 542edd362cbedef09e275e314f31c8a580f9901d
Author: ElianaRose <erw2138@barnard.edu>
Date:   Tue Nov 1 14:46:34 2016 -0400

    added pipeline files

commit dacfdcc1982eb2da4f647c6c8e3f2e65d9407453
Author: ElianaRose <erw2138@barnard.edu>
Date:   Wed Oct 26 17:56:50 2016 -0400

    added link to ShareLatex

commit d98874d7c571ece7c2d955750273a34d8bf9e8eb
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Wed Oct 26 17:06:13 2016 -0400

    Disallowed function parameters from being void type

commit e67bb6892d2350359a84ba861af56a5d4d300761
```

```
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Tue Oct 25 11:41:10 2016 -0400

    Minor additions including expr -> ID

commit 2f99ed22457529d666df1cf35c90338bcde09686
Author: ElianaRose <erw2138@barnard.edu>
Date:   Mon Oct 24 17:34:26 2016 -0400

    small fixes

commit e3c66162c1c55fb5a6f5e2c6b31a8d86250b1959
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Oct 23 22:37:44 2016 -0400

    updated changelog

commit 32e943a75c3495bb781d9a81f88293611adf8f3e
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Oct 23 14:32:58 2016 -0400

    added logo

commit 95203bf73e1ccff57de1287efd9fde00d0ae3bbb
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Oct 23 14:28:10 2016 -0400

    fixed formatting

commit 0da3de239eb1038246ab749171853e1c6b0491fc
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Oct 23 14:23:03 2016 -0400

    updated changelog

commit 1f5f4032bab86ea9c628fc990bfafba4d1cc81ea
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Oct 23 14:22:18 2016 -0400

    changed tensor sytax to T_{1, 2, 3}

commit 9fb531998359085ebbfdb52ee767b52d3311ac86
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Oct 23 14:06:38 2016 -0400

    updated changelog

commit 242b4cfa5ce9f2f3077635bb06a193ead63e754c
Author: ElianaRose <erw2138@barnard.edu>
Date:   Sun Oct 23 14:03:36 2016 -0400
```

```
    removed extra LET token, uneeded '|' and vdecl_list. No shift/reduce errors

commit 8ac31e7f88eabc3115238751d806525126f68bcd
Author: dyschwartz <ds3263@columbia.edu>
Date:   Sun Oct 23 11:48:45 2016 -0400

    added 3 ideas to change lines

commit a49f61bdfc791582e1d879215a494f34acf280ee
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sat Oct 22 17:43:37 2016 -0400

    Typo in changelog. Sorry, couldn't just leave it there.

commit 66243d789221af59ee03882249c44f32438d430e
Author: Elsbeth Turcan <eturcan@cs.columbia.edu>
Date:   Sat Oct 22 17:38:49 2016 -0400

    Enhanced tensor type (see changelog), removed 14 shift/reduce conflicts

commit 40ba6735cd6e874adc8dfa7555027eea1af68b99
Author: mr3522 <mr3522@columbia.edu>
Date:   Thu Oct 20 22:55:37 2016 -0400

    I think it's working.  15 shift-reduce.

commit 6793b9d33ace92000978f061452e31f0f54cc90b
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Oct 20 19:46:17 2016 -0400

    work in progress

commit c6abde1fe1f54f99449a5c3962dd2f8ec7300691
Merge: 4bfb338 5ba4adb
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Oct 20 15:18:50 2016 -0400

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS

commit 4bfb338a2f05b4e73864d1bea069b00fce29f388
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Oct 20 15:18:13 2016 -0400

    scanner working

commit 5ba4adbd8dd150520303c40add695a2543a7b4a6
Merge: 676a29e dcb25fd
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Thu Oct 13 17:16:19 2016 -0400

    Merge branch 'master' of https://github.com/dyschwartz/LaTenS
```

```
commit 676a29e81afb619e77536886a37a42529c64ca09
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Thu Oct 13 17:16:08 2016 -0400

    README

commit dcb25fd8f24e718996f77b601cfa101b367a8f7a
Author: Mohit Rajpal <mr3522@columbia.edu>
Date:   Thu Oct 13 16:51:47 2016 -0400

    fixed sedwards silly use of comments in ocamlyacc

commit e915683a13060bee331c04233dcd153542baf361
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Thu Oct 13 16:41:32 2016 -0400

    micro c files copied from slides

commit ef70924a648d411cf2d927ab1900a5e8c47dae86
Author: Daniel Schwartz <ds3263@columbia.edu>
Date:   Thu Oct 6 16:32:40 2016 -0400

    first commit
```

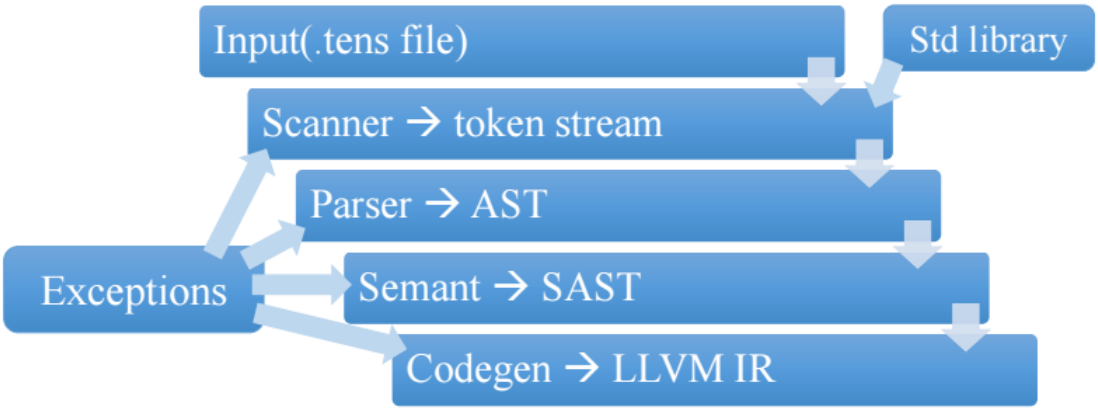
6 Architectural Design

The LaTenS compiler is a 4 pass non-optimizing, non-linking compiler with LLVM IR as the target language. The lexer and parser are implemented with OcamlLex and OcamlYacc respectively. Afterwards, a static type checking and enforcement pass is conducted followed by a code generation pass. The compiler uses an abstract syntax tree as the sole intermediate representation. The compiler is implemented in the following files:

- scanner.mll: The lexer for the compiler.
- parser.mly: The context free language definitions for LaTenS.
- ast.ml: Abstract Syntax Tree IR type definitions.
- semant.ml: Static type and semantic checker which generates a typed Abstract Syntax Tree.
- sast.ml: Semantically checked typed abstract syntax tree type definitions.
- exceptions.ml: Exception definitions for human readable error printing.
- codegen.ml: LLVM-IR code generator.
- latens.ml: Top level executable implementing the 4-pass LaTenS compiler.

An illustration of the components is presented.
The nontrivial subcomponents of the compiler are described herein.

Figure 1: A block diagram of the LaTenS compiler



6.1 Semantic Checking

The semantic checker transforms an AST IR into a SAST IR, a Semantically Checked Abstract Syntax Tree where each applicable node in the tree is given a type. The following conditions are checked, and the following information is calculated, in the semantic checker:

- When a new identifier is declared, that identifier must not already exist.
- If an identifier is used, that identifier must already be visible in the current scope according to our static scoping rules (any identifier declared in a parent or ancestor scope is accessible, where child scopes are generated at each function, block, if, for, and while).
- The types of arguments sent to a binary or unary operator or assignment are valid both (1) in that operator and (2) with one another in that operator.
- Any expression inside a global assignment or a tensor index or element is a valid and allowed expression. For example, an expression which evaluates to string is not a valid tensor element.
- Semantic checking sends the type of all expressions along to code generation from the bottom up (for example, a float multiplied with an integer is considered a float). The types of all identifiers are also inferred.
- Any tensor literal is a valid tensor literal; that is, every dimension of the tensor is the same size and every sub-tensor at the same level is of the same rank and size.
- Any function call uses a previously-defined function name or one of the built-in functions, and the arguments sent to that functions are of the correct types.
- Any tensor multiplication is valid. Semantic checking also collects the bound indices for code generation and calculates the rank of the resulting tensor.
- Every statement type's specific conditions are satisfied. For example, an if statement includes one expression that evaluates to an integer (that is, a boolean condition) and two statement blocks (the if and else blocks, either of which may be empty).
- Every function either (1) returns an expression of the type it was declared to return, or (2) does not return anything if it was declared to be void.

Once it is finished, the semantic checker has produced an SAST, which includes a list of global variables as well as a tree consisting of semantically checked nodes for each function.

6.2 Code generation

The code generation pass, included in codegen.ml, transforms a semantically checked abstract syntax tree into LLVM IR. Code generation follows a straightforward post order traversal to generate LLVM primitives akin to three-address code generation. The initial code generation code was borrowed from the implementation of Micro-C. Indeed many of our primitive features are identical to Micro-C. Code generation using LLVM is accomplished with the OCaml bindings for LLVM. The bindings enable generation of arbitrary LLVM constructs at defined positions in the flat LLVM-IR file. An LLVM file can be thought of as a collection of basic blocks connected with conditional or unconditional branches. Each basic block holds the following properties:

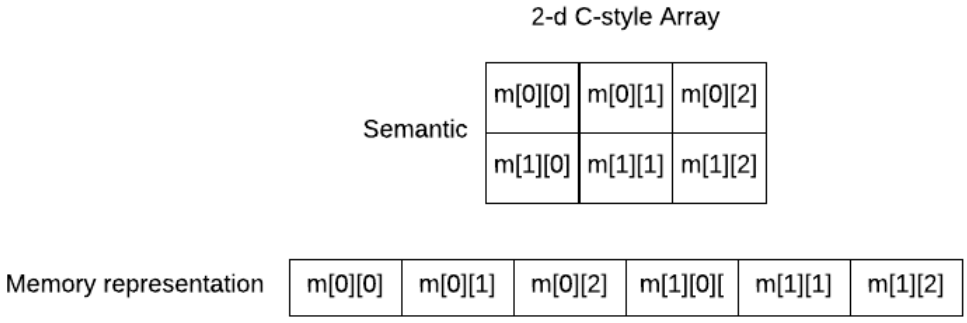
- The only entry point to a basic block is the first statement in the basic block.
- Each basic block must be terminated with an unconditional jump.

These enforced paradigms enable LLVM-IR to be a structured high level assembly language. Furthermore, LLVM-IR is strongly typed which catches many subtle errors early in the development process. The author notes that compiler generated LLVM-IR is often bloated due to unnecessary casts and is difficult to read for any non-trivial program. The author often compiled LLVM-IR to x86_64 assembly for ease of readability and comprehension.
Many features in our language are trivial to implement using LLVM-IR, and Micro-C serves as the perfect reference of the high level structure of LaTenS. The notable features of LaTenS which distinguish the language from Micro-C are presented below.

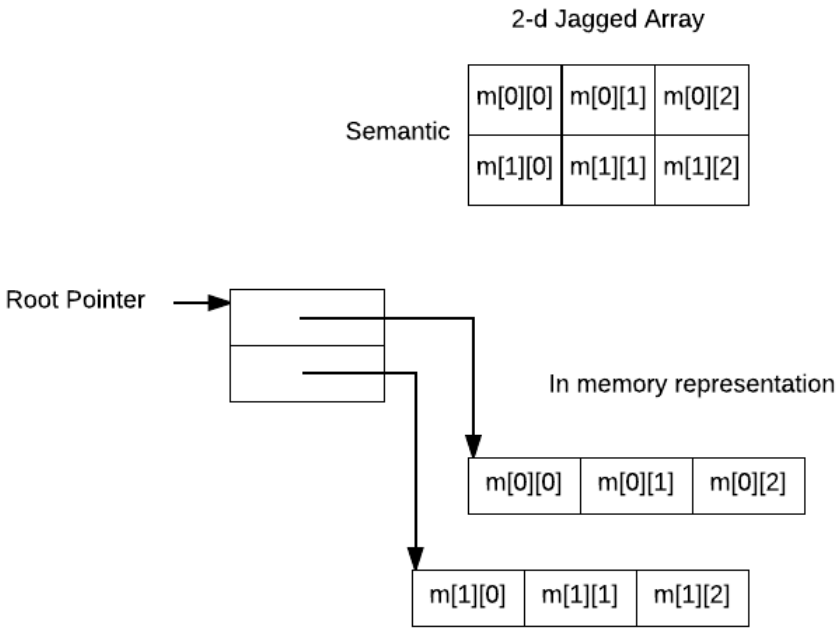
6.3 Notable Features

6.3.1 Tensors

Tensors are a generalization of matrices. Thus in terms of code tensors are n-d arrays. Typically there are two schools of thought on the implementation of n-d arrays. The C-style of multidimensional arrays involve allocating large chunks of memory which are then indexed using the known size of each rank of the array. This requires knowledge of the dimension of each rank prior to allocation. An illustration of C-style arrays is presented:



Jagged arrays are multidimensional arrays implemented as trees connected with pointers.
We implemented Tensors as jagged arrays. We believed the highly recursive nature of Ocaml would lend itself well to processing a recursive data structure such as trees. However, there is one small caveat.
Note that typical Jagged arrays contain values at the terminal level. In our implementation we chose to fill the final level with pointers to floating point values. This allowed us to easily take reference to portions of a tensor by copying the relevant pointers. Overall this functionality was used extensively in slice notation.



6.3.2 Code expansion

Almost all tensor expressions involve code expansions. Tensor allocation, slicing, and binary operations involve code expansion. Our code expansion scheme is non-trivial as the constructs required are not valid LaTenS statements or expressions.
Tensor allocation requires nested for loops with heap allocation calls for each level of the jagged array tree. Typically a heap allocation for a 2-d array is as follows

```
float **arr2d;
arr2d = malloc(size(float) * 2);
for(i = 0; i < 2; i = i + 1)
{
    arr2d[i] = malloc(size(float) * 5);
}
```

We note that the constructs for for-loops exists in LaTenS, but subindexing and heap allocation do not. These issues were solved in Codegen using a new statement type internal to LaTenS: A Callback statement. A callback statement containsis a one argument function which takes a basic block location, and returns a basic block location. However, the callback is free to add arbitrary code and create/append basic blocks to accomplish its task.
The code generation programming paradigm uses LaTenS statements to generate flow control code such as loops, but uses internal Callback statements to generate arbitrary internal code constructs. These constructs are typically in the form of heap allocation, dimension or rank calculations, tensor copying, and binary operation logic.
The statement builder, when encountering a callback, calls the associated function to fill in code for which no equivalent LaTenS construct exists. Typically, this callback itself will fill in some LLVM-IR, and then call the statement builder to do more work.
We note that the callback function takes one argument, so one may think it cannot do any interesting work. However, with function currying, an arbitrarily complex callback function can be constructed. The typical

callback function generates a few line of LLVM-IR. However it then constructs an AST node, which contains a callback to itself, and passes this to the statement builder. This mutually recursive, ping-pong, approach allows the tensor generation code to leverage existing LaTenS constructs.

7 Test Plan

7.1 Source Code and Generated Target Code

We demonstrate the LaTenS and LLVM code of our three demo programs. We also implemented a LaTeX pretty-printer which allows a user to pretty-print their code into a LaTeX file.

7.1.1 Example: Cattle Age Distribution Demo

The first demo used a Leslie matrix to show the age distribution of purchased cattle over a 42-year period, assuming that a person buys 40 head of 2-year-old cattle.
LaTenS source code:

```
%% A pithy code sample that shows us
%% how much cattle we would have if we
%% bought 40 cattle some number of years ago.
%% With the help of our friends: https://www.wku.edu/mathmatters/documents/mathmattersep13.pdf
%% (ie we implemented their stuff in LaTenS)
int main () {
    print("Suppose you purchase 40 head of 2 year old cattle\n");

    %% Our Leslie Matrix, for heads of cattle
    let L = [[ 0, .5, 1.1, 1.3, 0.8, .4],
               [.7, 0, 0, 0, 0, 0],
               [0, .9, 0, 0, 0, 0],
               [0, 0, .8, 0, 0, 0],
               [0, 0, 0, .7, 0, 0],
               [0, 0, 0, 0, .3, 0]];

    %% We've got 40 cattle at t=0; where t is measured in years
    let D0 = [0, 40, 0, 0, 0, 0];


    print("given survival rates and birth rates: \n");
    print(L);


    print("you want to know what your age distribution is\n");
    print("after 2 years: ");

    %% Each multiplication moves t forward 2 years.
    %% So now t=2
    let D = L_{a, b}*D0_{b};
    print(D);


    %% Let's jump t forward 28 more years
    %% So t = 2 + 28 = 30
    for(let i = 0; i < 14; i = i+1){
      D = L_{a, b}*D_{b};
    }
```

```
    print("after 30 years: ");
    print(D);


    %% And of course, no demo would be complete
    %% without showing t=42
    for(let i = 0; i < 26; i = i+1){
      D = L_{a, b}*D_{b};
    }
    print("after 42 years: ");
    print(D);

    return 0;
}
```

Using the command `./latens.native demos/demo1.tens`, we compile our code into LLVM:

```
; ModuleID = 'LaTenS'

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt = private unnamed_addr constant [5 x i8] c"%.6f\00"
@"Suppose you purchase 40 head of 2 year old cattle\0A" = private unnamed_addr constant [51 x i8]
    c"Suppose you purchase 40 head of 2 year old cattle\0A\00"
@"given survival rates and birth rates: \0A" = private unnamed_addr constant [40 x i8] c"given
    survival rates and birth rates: \0A\00"
@"you want to know what your age distribution is\0A" = private unnamed_addr constant [48 x i8]
    c"you want to know what your age distribution is\0A\00"
@"after 2 years: " = private unnamed_addr constant [16 x i8] c"after 2 years: \00"
@"after 30 years: " = private unnamed_addr constant [17 x i8] c"after 30 years: \00"
@"after 42 years: " = private unnamed_addr constant [17 x i8] c"after 42 years: \00"
@fmt.1 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.2 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.3 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.4 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.5 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.6 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.7 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.8 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.9 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.10 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.11 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.12 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.13 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.14 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.15 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.16 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.17 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.18 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.19 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.20 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.21 = private unnamed_addr constant [5 x i8] c"%.6f\00"
```

```
@fmt.22 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.23 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.24 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.25 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.26 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.27 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.28 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.29 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.30 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.31 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.32 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.33 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.34 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.35 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.36 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.37 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.38 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.39 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.40 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.41 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.42 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.43 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.44 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.45 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.46 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.47 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.48 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.49 = private unnamed_addr constant [5 x i8] c"%.6f\00"

declare i64 @printf(i8*, ...)

declare i32 @strlen(i8*)

declare i64 @sprintf(i8*, i8*, ...)

declare i64 @atoi(i8*)

declare double @atof(i8*)

declare i64 @fopen(i8*, i8*)

declare i64 @fclose(i64)

declare i32 @fread(i64*, i32, i32, i64*)

declare i64 @fwrite(i64*, i32, i32, i64*)

declare double @sqrt(double)

define i64 @main() {
entry:
  %printf_ignore = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([51 x i8], [51 x i8]*
      @"Suppose you purchase 40 head of 2 year old cattle\0A", i32 0, i32 0))
  %L = alloca i64*
  %0 = trunc i64 4 to i32
  %mallocsize = mul i32 %0, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
```

```
92    %malloccall = tail call i8* @malloc(i32 %mallocsize)
93    %immediate_tmp = bitcast i8* %malloccall to i64*
94    store i64 2, i64* %immediate_tmp
95    %intcast = ptrtoint i64* %immediate_tmp to i64
96    %pmath = add i64 %intcast, 8
97    %pointercast = inttoptr i64 %pmath to i64*
98    store i64 6, i64* %pointercast
99    %intcast1 = ptrtoint i64* %immediate_tmp to i64
100   %pmath2 = add i64 %intcast1, 16
101   %pointercast3 = inttoptr i64 %pmath2 to i64*
102   store i64 6, i64* %pointercast3
103   %intcast4 = ptrtoint i64* %immediate_tmp to i64
104   %pmath5 = add i64 %intcast4, 24
105   %pointercast6 = inttoptr i64 %pmath5 to i64*
106   %1 = trunc i64 6 to i32
107   %mallocsize7 = mul i32 %1, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
108   %malloccall8 = tail call i8* @malloc(i32 %mallocsize7)
109   %fstarstorage = bitcast i8* %malloccall8 to i64*
110   %2 = trunc i64 6 to i32
111   %mallocsize9 = mul i32 %2, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
112   %malloccall10 = tail call i8* @malloc(i32 %mallocsize9)
113   %fstorage = bitcast i8* %malloccall10 to i64*
114   %intcast11 = ptrtoint i64* %fstorage to i64
115   %pmath12 = add i64 %intcast11, 0
116   %pointercast13 = inttoptr i64 %pmath12 to i64*
117   %malloccall14 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
          i32 1) to i32))
118   %factual = bitcast i8* %malloccall14 to double*
119   %ptwtointforfpstorage = ptrtoint double* %factual to i64
120   store double 0.000000e+00, double* %factual
121   store i64 %ptwtointforfpstorage, i64* %pointercast13
122   %intcast15 = ptrtoint i64* %fstorage to i64
123   %pmath16 = add i64 %intcast15, 8
124   %pointercast17 = inttoptr i64 %pmath16 to i64*
125   %malloccall18 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
          i32 1) to i32))
126   %factual19 = bitcast i8* %malloccall18 to double*
127   %ptwtointforfpstorage20 = ptrtoint double* %factual19 to i64
128   store double 5.000000e-01, double* %factual19
129   store i64 %ptwtointforfpstorage20, i64* %pointercast17
130   %intcast21 = ptrtoint i64* %fstorage to i64
131   %pmath22 = add i64 %intcast21, 16
132   %pointercast23 = inttoptr i64 %pmath22 to i64*
133   %malloccall24 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
          i32 1) to i32))
134   %factual25 = bitcast i8* %malloccall24 to double*
135   %ptwtointforfpstorage26 = ptrtoint double* %factual25 to i64
136   store double 1.100000e+00, double* %factual25
137   store i64 %ptwtointforfpstorage26, i64* %pointercast23
138   %intcast27 = ptrtoint i64* %fstorage to i64
139   %pmath28 = add i64 %intcast27, 24
140   %pointercast29 = inttoptr i64 %pmath28 to i64*
141   %malloccall30 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
          i32 1) to i32))
142   %factual31 = bitcast i8* %malloccall30 to double*
143   %ptwtointforfpstorage32 = ptrtoint double* %factual31 to i64
```

```
    store double 1.300000e+00, double* %factual31
    store i64 %ptwtointforfpstorage32, i64* %pointercast29
    %intcast33 = ptrtoint i64* %fstorage to i64
    %pmath34 = add i64 %intcast33, 32
    %pointercast35 = inttoptr i64 %pmath34 to i64*
    %malloccall36 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
        i32 1) to i32))
    %factual37 = bitcast i8* %malloccall36 to double*
    %ptwtointforfpstorage38 = ptrtoint double* %factual37 to i64
    store double 8.000000e-01, double* %factual37
    store i64 %ptwtointforfpstorage38, i64* %pointercast35
    %intcast39 = ptrtoint i64* %fstorage to i64
    %pmath40 = add i64 %intcast39, 40
    %pointercast41 = inttoptr i64 %pmath40 to i64*
    %malloccall42 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
        i32 1) to i32))
    %factual43 = bitcast i8* %malloccall42 to double*
    %ptwtointforfpstorage44 = ptrtoint double* %factual43 to i64
    store double 4.000000e-01, double* %factual43
    store i64 %ptwtointforfpstorage44, i64* %pointercast41
    %ptrtointfornlstorage = ptrtoint i64* %fstorage to i64
    store i64 %ptrtointfornlstorage, i64* %fstarstorage
    %ptwtointfornlstorage = ptrtoint i64* %fstarstorage to i64
    %nextptr = add i64 %ptwtointfornlstorage, 8
    %nextptrptr = inttoptr i64 %nextptr to i64*
    %3 = trunc i64 6 to i32
    %mallocsize45 = mul i32 %3, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
    %malloccall46 = tail call i8* @malloc(i32 %mallocsize45)
    %fstorage47 = bitcast i8* %malloccall46 to i64*
    %intcast48 = ptrtoint i64* %fstorage47 to i64
    %pmath49 = add i64 %intcast48, 0
    %pointercast50 = inttoptr i64 %pmath49 to i64*
    %malloccall51 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
        i32 1) to i32))
    %factual52 = bitcast i8* %malloccall51 to double*
    %ptwtointforfpstorage53 = ptrtoint double* %factual52 to i64
    store double 7.000000e-01, double* %factual52
    store i64 %ptwtointforfpstorage53, i64* %pointercast50
    %intcast54 = ptrtoint i64* %fstorage47 to i64
    %pmath55 = add i64 %intcast54, 8
    %pointercast56 = inttoptr i64 %pmath55 to i64*
    %malloccall57 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
        i32 1) to i32))
    %factual58 = bitcast i8* %malloccall57 to double*
    %ptwtointforfpstorage59 = ptrtoint double* %factual58 to i64
    store double 0.000000e+00, double* %factual58
    store i64 %ptwtointforfpstorage59, i64* %pointercast56
    %intcast60 = ptrtoint i64* %fstorage47 to i64
    %pmath61 = add i64 %intcast60, 16
    %pointercast62 = inttoptr i64 %pmath61 to i64*
    %malloccall63 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
        i32 1) to i32))
    %factual64 = bitcast i8* %malloccall63 to double*
    %ptwtointforfpstorage65 = ptrtoint double* %factual64 to i64
    store double 0.000000e+00, double* %factual64
    store i64 %ptwtointforfpstorage65, i64* %pointercast62
```

```
  %intcast66 = ptrtoint i64* %fstorage47 to i64
  %pmath67 = add i64 %intcast66, 24
  %pointercast68 = inttoptr i64 %pmath67 to i64*
  %malloccall69 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
      i32 1) to i32))
  %factual70 = bitcast i8* %malloccall69 to double*
  %ptwtointforfpstorage71 = ptrtoint double* %factual70 to i64
  store double 0.000000e+00, double* %factual70
  store i64 %ptwtointforfpstorage71, i64* %pointercast68
  %intcast72 = ptrtoint i64* %fstorage47 to i64
  %pmath73 = add i64 %intcast72, 32
  %pointercast74 = inttoptr i64 %pmath73 to i64*
  %malloccall75 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
      i32 1) to i32))
  %factual76 = bitcast i8* %malloccall75 to double*
  %ptwtointforfpstorage77 = ptrtoint double* %factual76 to i64
  store double 0.000000e+00, double* %factual76
  store i64 %ptwtointforfpstorage77, i64* %pointercast74
  %intcast78 = ptrtoint i64* %fstorage47 to i64
  %pmath79 = add i64 %intcast78, 40
  %pointercast80 = inttoptr i64 %pmath79 to i64*
  %malloccall81 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
      i32 1) to i32))
  %factual82 = bitcast i8* %malloccall81 to double*
  %ptwtointforfpstorage83 = ptrtoint double* %factual82 to i64
  store double 0.000000e+00, double* %factual82
  store i64 %ptwtointforfpstorage83, i64* %pointercast80
  %ptrtointfornlstorage84 = ptrtoint i64* %fstorage47 to i64
  store i64 %ptrtointfornlstorage84, i64* %nextptrptr
  %ptwtointfornlstorage85 = ptrtoint i64* %nextptrptr to i64
  %nextptr86 = add i64 %ptwtointfornlstorage85, 8
  %nextptrptr87 = inttoptr i64 %nextptr86 to i64*
  %4 = trunc i64 6 to i32
  %mallocsize88 = mul i32 %4, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall89 = tail call i8* @malloc(i32 %mallocsize88)
  %fstorage90 = bitcast i8* %malloccall89 to i64*
  %intcast91 = ptrtoint i64* %fstorage90 to i64
  %pmath92 = add i64 %intcast91, 0
  %pointercast93 = inttoptr i64 %pmath92 to i64*
  %malloccall94 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
      i32 1) to i32))
  %factual95 = bitcast i8* %malloccall94 to double*
  %ptwtointforfpstorage96 = ptrtoint double* %factual95 to i64
  store double 0.000000e+00, double* %factual95
  store i64 %ptwtointforfpstorage96, i64* %pointercast93
  %intcast97 = ptrtoint i64* %fstorage90 to i64
  %pmath98 = add i64 %intcast97, 8
  %pointercast99 = inttoptr i64 %pmath98 to i64*
  %malloccall100 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual101 = bitcast i8* %malloccall100 to double*
  %ptwtointforfpstorage102 = ptrtoint double* %factual101 to i64
  store double 9.000000e-01, double* %factual101
  store i64 %ptwtointforfpstorage102, i64* %pointercast99
  %intcast103 = ptrtoint i64* %fstorage90 to i64
  %pmath104 = add i64 %intcast103, 16
```

```
246   %pointercast105 = inttoptr i64 %pmath104 to i64*
247   %malloccall106 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
248   %factual107 = bitcast i8* %malloccall106 to double*
249   %ptwtointforfpstorage108 = ptrtoint double* %factual107 to i64
250   store double 0.000000e+00, double* %factual107
251   store i64 %ptwtointforfpstorage108, i64* %pointercast105
252   %intcast109 = ptrtoint i64* %fstorage90 to i64
253   %pmath110 = add i64 %intcast109, 24
254   %pointercast111 = inttoptr i64 %pmath110 to i64*
255   %malloccall112 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
256   %factual113 = bitcast i8* %malloccall112 to double*
257   %ptwtointforfpstorage114 = ptrtoint double* %factual113 to i64
258   store double 0.000000e+00, double* %factual113
259   store i64 %ptwtointforfpstorage114, i64* %pointercast111
260   %intcast115 = ptrtoint i64* %fstorage90 to i64
261   %pmath116 = add i64 %intcast115, 32
262   %pointercast117 = inttoptr i64 %pmath116 to i64*
263   %malloccall118 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
264   %factual119 = bitcast i8* %malloccall118 to double*
265   %ptwtointforfpstorage120 = ptrtoint double* %factual119 to i64
266   store double 0.000000e+00, double* %factual119
267   store i64 %ptwtointforfpstorage120, i64* %pointercast117
268   %intcast121 = ptrtoint i64* %fstorage90 to i64
269   %pmath122 = add i64 %intcast121, 40
270   %pointercast123 = inttoptr i64 %pmath122 to i64*
271   %malloccall124 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
272   %factual125 = bitcast i8* %malloccall124 to double*
273   %ptwtointforfpstorage126 = ptrtoint double* %factual125 to i64
274   store double 0.000000e+00, double* %factual125
275   store i64 %ptwtointforfpstorage126, i64* %pointercast123
276   %ptrtointfornlstorage127 = ptrtoint i64* %fstorage90 to i64
277   store i64 %ptrtointfornlstorage127, i64* %nextptrptr87
278   %ptwtointfornlstorage128 = ptrtoint i64* %nextptrptr87 to i64
279   %nextptr129 = add i64 %ptwtointfornlstorage128, 8
280   %nextptrptr130 = inttoptr i64 %nextptr129 to i64*
281   %5 = trunc i64 6 to i32
282   %mallocsize131 = mul i32 %5, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
283   %malloccall132 = tail call i8* @malloc(i32 %mallocsize131)
284   %fstorage133 = bitcast i8* %malloccall132 to i64*
285   %intcast134 = ptrtoint i64* %fstorage133 to i64
286   %pmath135 = add i64 %intcast134, 0
287   %pointercast136 = inttoptr i64 %pmath135 to i64*
288   %malloccall137 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
289   %factual138 = bitcast i8* %malloccall137 to double*
290   %ptwtointforfpstorage139 = ptrtoint double* %factual138 to i64
291   store double 0.000000e+00, double* %factual138
292   store i64 %ptwtointforfpstorage139, i64* %pointercast136
293   %intcast140 = ptrtoint i64* %fstorage133 to i64
294   %pmath141 = add i64 %intcast140, 8
295   %pointercast142 = inttoptr i64 %pmath141 to i64*
296   %malloccall143 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
```

```
        null, i32 1) to i32))
  %factual144 = bitcast i8* %malloccall143 to double*
  %ptwtointforfpstorage145 = ptrtoint double* %factual144 to i64
  store double 0.000000e+00, double* %factual144
  store i64 %ptwtointforfpstorage145, i64* %pointercast142
  %intcast146 = ptrtoint i64* %fstorage133 to i64
  %pmath147 = add i64 %intcast146, 16
  %pointercast148 = inttoptr i64 %pmath147 to i64*
  %malloccall149 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual150 = bitcast i8* %malloccall149 to double*
  %ptwtointforfpstorage151 = ptrtoint double* %factual150 to i64
  store double 8.000000e-01, double* %factual150
  store i64 %ptwtointforfpstorage151, i64* %pointercast148
  %intcast152 = ptrtoint i64* %fstorage133 to i64
  %pmath153 = add i64 %intcast152, 24
  %pointercast154 = inttoptr i64 %pmath153 to i64*
  %malloccall155 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual156 = bitcast i8* %malloccall155 to double*
  %ptwtointforfpstorage157 = ptrtoint double* %factual156 to i64
  store double 0.000000e+00, double* %factual156
  store i64 %ptwtointforfpstorage157, i64* %pointercast154
  %intcast158 = ptrtoint i64* %fstorage133 to i64
  %pmath159 = add i64 %intcast158, 32
  %pointercast160 = inttoptr i64 %pmath159 to i64*
  %malloccall161 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual162 = bitcast i8* %malloccall161 to double*
  %ptwtointforfpstorage163 = ptrtoint double* %factual162 to i64
  store double 0.000000e+00, double* %factual162
  store i64 %ptwtointforfpstorage163, i64* %pointercast160
  %intcast164 = ptrtoint i64* %fstorage133 to i64
  %pmath165 = add i64 %intcast164, 40
  %pointercast166 = inttoptr i64 %pmath165 to i64*
  %malloccall167 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual168 = bitcast i8* %malloccall167 to double*
  %ptwtointforfpstorage169 = ptrtoint double* %factual168 to i64
  store double 0.000000e+00, double* %factual168
  store i64 %ptwtointforfpstorage169, i64* %pointercast166
  %ptrtointfornlstorage170 = ptrtoint i64* %fstorage133 to i64
  store i64 %ptrtointfornlstorage170, i64* %nextptrptr130
  %ptwtointfornlstorage171 = ptrtoint i64* %nextptrptr130 to i64
  %nextptr172 = add i64 %ptwtointfornlstorage171, 8
  %nextptrptr173 = inttoptr i64 %nextptr172 to i64*
  %6 = trunc i64 6 to i32
  %mallocsize174 = mul i32 %6, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall175 = tail call i8* @malloc(i32 %mallocsize174)
  %fstorage176 = bitcast i8* %malloccall175 to i64*
  %intcast177 = ptrtoint i64* %fstorage176 to i64
  %pmath178 = add i64 %intcast177, 0
  %pointercast179 = inttoptr i64 %pmath178 to i64*
  %malloccall180 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual181 = bitcast i8* %malloccall180 to double*
```

```
347   %ptwtointforfpstorage182 = ptrtoint double* %factual181 to i64
348   store double 0.000000e+00, double* %factual181
349   store i64 %ptwtointforfpstorage182, i64* %pointercast179
350   %intcast183 = ptrtoint i64* %fstorage176 to i64
351   %pmath184 = add i64 %intcast183, 8
352   %pointercast185 = inttoptr i64 %pmath184 to i64*
353   %malloccall186 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
354   %factual187 = bitcast i8* %malloccall186 to double*
355   %ptwtointforfpstorage188 = ptrtoint double* %factual187 to i64
356   store double 0.000000e+00, double* %factual187
357   store i64 %ptwtointforfpstorage188, i64* %pointercast185
358   %intcast189 = ptrtoint i64* %fstorage176 to i64
359   %pmath190 = add i64 %intcast189, 16
360   %pointercast191 = inttoptr i64 %pmath190 to i64*
361   %malloccall192 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
362   %factual193 = bitcast i8* %malloccall192 to double*
363   %ptwtointforfpstorage194 = ptrtoint double* %factual193 to i64
364   store double 0.000000e+00, double* %factual193
365   store i64 %ptwtointforfpstorage194, i64* %pointercast191
366   %intcast195 = ptrtoint i64* %fstorage176 to i64
367   %pmath196 = add i64 %intcast195, 24
368   %pointercast197 = inttoptr i64 %pmath196 to i64*
369   %malloccall198 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
370   %factual199 = bitcast i8* %malloccall198 to double*
371   %ptwtointforfpstorage200 = ptrtoint double* %factual199 to i64
372   store double 7.000000e-01, double* %factual199
373   store i64 %ptwtointforfpstorage200, i64* %pointercast197
374   %intcast201 = ptrtoint i64* %fstorage176 to i64
375   %pmath202 = add i64 %intcast201, 32
376   %pointercast203 = inttoptr i64 %pmath202 to i64*
377   %malloccall204 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
378   %factual205 = bitcast i8* %malloccall204 to double*
379   %ptwtointforfpstorage206 = ptrtoint double* %factual205 to i64
380   store double 0.000000e+00, double* %factual205
381   store i64 %ptwtointforfpstorage206, i64* %pointercast203
382   %intcast207 = ptrtoint i64* %fstorage176 to i64
383   %pmath208 = add i64 %intcast207, 40
384   %pointercast209 = inttoptr i64 %pmath208 to i64*
385   %malloccall210 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
386   %factual211 = bitcast i8* %malloccall210 to double*
387   %ptwtointforfpstorage212 = ptrtoint double* %factual211 to i64
388   store double 0.000000e+00, double* %factual211
389   store i64 %ptwtointforfpstorage212, i64* %pointercast209
390   %ptrtointfornlstorage213 = ptrtoint i64* %fstorage176 to i64
391   store i64 %ptrtointfornlstorage213, i64* %nextptrptr173
392   %ptwtointfornlstorage214 = ptrtoint i64* %nextptrptr173 to i64
393   %nextptr215 = add i64 %ptwtointfornlstorage214, 8
394   %nextptrptr216 = inttoptr i64 %nextptr215 to i64*
395   %7 = trunc i64 6 to i32
396   %mallocsize217 = mul i32 %7, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
397   %malloccall218 = tail call i8* @malloc(i32 %mallocsize217)
```

```
  %fstorage219 = bitcast i8* %malloccall218 to i64*
  %intcast220 = ptrtoint i64* %fstorage219 to i64
  %pmath221 = add i64 %intcast220, 0
  %pointercast222 = inttoptr i64 %pmath221 to i64*
  %malloccall223 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual224 = bitcast i8* %malloccall223 to double*
  %ptwtointforfpstorage225 = ptrtoint double* %factual224 to i64
  store double 0.000000e+00, double* %factual224
  store i64 %ptwtointforfpstorage225, i64* %pointercast222
  %intcast226 = ptrtoint i64* %fstorage219 to i64
  %pmath227 = add i64 %intcast226, 8
  %pointercast228 = inttoptr i64 %pmath227 to i64*
  %malloccall229 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual230 = bitcast i8* %malloccall229 to double*
  %ptwtointforfpstorage231 = ptrtoint double* %factual230 to i64
  store double 0.000000e+00, double* %factual230
  store i64 %ptwtointforfpstorage231, i64* %pointercast228
  %intcast232 = ptrtoint i64* %fstorage219 to i64
  %pmath233 = add i64 %intcast232, 16
  %pointercast234 = inttoptr i64 %pmath233 to i64*
  %malloccall235 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual236 = bitcast i8* %malloccall235 to double*
  %ptwtointforfpstorage237 = ptrtoint double* %factual236 to i64
  store double 0.000000e+00, double* %factual236
  store i64 %ptwtointforfpstorage237, i64* %pointercast234
  %intcast238 = ptrtoint i64* %fstorage219 to i64
  %pmath239 = add i64 %intcast238, 24
  %pointercast240 = inttoptr i64 %pmath239 to i64*
  %malloccall241 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual242 = bitcast i8* %malloccall241 to double*
  %ptwtointforfpstorage243 = ptrtoint double* %factual242 to i64
  store double 0.000000e+00, double* %factual242
  store i64 %ptwtointforfpstorage243, i64* %pointercast240
  %intcast244 = ptrtoint i64* %fstorage219 to i64
  %pmath245 = add i64 %intcast244, 32
  %pointercast246 = inttoptr i64 %pmath245 to i64*
  %malloccall247 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual248 = bitcast i8* %malloccall247 to double*
  %ptwtointforfpstorage249 = ptrtoint double* %factual248 to i64
  store double 3.000000e-01, double* %factual248
  store i64 %ptwtointforfpstorage249, i64* %pointercast246
  %intcast250 = ptrtoint i64* %fstorage219 to i64
  %pmath251 = add i64 %intcast250, 40
  %pointercast252 = inttoptr i64 %pmath251 to i64*
  %malloccall253 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual254 = bitcast i8* %malloccall253 to double*
  %ptwtointforfpstorage255 = ptrtoint double* %factual254 to i64
  store double 0.000000e+00, double* %factual254
  store i64 %ptwtointforfpstorage255, i64* %pointercast252
  %ptrtointfornlstorage256 = ptrtoint i64* %fstorage219 to i64
```

```
448   store i64 %ptrtointfornlstorage256, i64* %nextptrptr216
449   %ptwtointfornlstorage257 = ptrtoint i64* %nextptrptr216 to i64
450   %nextptr258 = add i64 %ptwtointfornlstorage257, 8
451   %nextptrptr259 = inttoptr i64 %nextptr258 to i64*
452   %ptrtointfornlstarstorage = ptrtoint i64* %fstarstorage to i64
453   store i64 %ptrtointfornlstarstorage, i64* %pointercast6
454   %ptwtointfornlstorage260 = ptrtoint i64* %pointercast6 to i64
455   %nextptr261 = add i64 %ptwtointfornlstorage260, 8
456   %nextptrptr262 = inttoptr i64 %nextptr261 to i64*
457   store i64* %immediate_tmp, i64** %L
458   %D0 = alloca i64*
459   %8 = trunc i64 3 to i32
460   %mallocsize263 = mul i32 %8, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
461   %malloccall264 = tail call i8* @malloc(i32 %mallocsize263)
462   %immediate_tmp265 = bitcast i8* %malloccall264 to i64*
463   store i64 1, i64* %immediate_tmp265
464   %intcast266 = ptrtoint i64* %immediate_tmp265 to i64
465   %pmath267 = add i64 %intcast266, 8
466   %pointercast268 = inttoptr i64 %pmath267 to i64*
467   store i64 6, i64* %pointercast268
468   %intcast269 = ptrtoint i64* %immediate_tmp265 to i64
469   %pmath270 = add i64 %intcast269, 16
470   %pointercast271 = inttoptr i64 %pmath270 to i64*
471   %9 = trunc i64 6 to i32
472   %mallocsize272 = mul i32 %9, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
473   %malloccall273 = tail call i8* @malloc(i32 %mallocsize272)
474   %fstorage274 = bitcast i8* %malloccall273 to i64*
475   %intcast275 = ptrtoint i64* %fstorage274 to i64
476   %pmath276 = add i64 %intcast275, 0
477   %pointercast277 = inttoptr i64 %pmath276 to i64*
478   %malloccall278 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
479   %factual279 = bitcast i8* %malloccall278 to double*
480   %ptwtointforfpstorage280 = ptrtoint double* %factual279 to i64
481   store double 0.000000e+00, double* %factual279
482   store i64 %ptwtointforfpstorage280, i64* %pointercast277
483   %intcast281 = ptrtoint i64* %fstorage274 to i64
484   %pmath282 = add i64 %intcast281, 8
485   %pointercast283 = inttoptr i64 %pmath282 to i64*
486   %malloccall284 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
487   %factual285 = bitcast i8* %malloccall284 to double*
488   %ptwtointforfpstorage286 = ptrtoint double* %factual285 to i64
489   store double 4.000000e+01, double* %factual285
490   store i64 %ptwtointforfpstorage286, i64* %pointercast283
491   %intcast287 = ptrtoint i64* %fstorage274 to i64
492   %pmath288 = add i64 %intcast287, 16
493   %pointercast289 = inttoptr i64 %pmath288 to i64*
494   %malloccall290 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
495   %factual291 = bitcast i8* %malloccall290 to double*
496   %ptwtointforfpstorage292 = ptrtoint double* %factual291 to i64
497   store double 0.000000e+00, double* %factual291
498   store i64 %ptwtointforfpstorage292, i64* %pointercast289
499   %intcast293 = ptrtoint i64* %fstorage274 to i64
500   %pmath294 = add i64 %intcast293, 24
```

```
501   %pointercast295 = inttoptr i64 %pmath294 to i64*
502   %malloccall296 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
503   %factual297 = bitcast i8* %malloccall296 to double*
504   %ptwtointforfpstorage298 = ptrtoint double* %factual297 to i64
505   store double 0.000000e+00, double* %factual297
506   store i64 %ptwtointforfpstorage298, i64* %pointercast295
507   %intcast299 = ptrtoint i64* %fstorage274 to i64
508   %pmath300 = add i64 %intcast299, 32
509   %pointercast301 = inttoptr i64 %pmath300 to i64*
510   %malloccall302 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
511   %factual303 = bitcast i8* %malloccall302 to double*
512   %ptwtointforfpstorage304 = ptrtoint double* %factual303 to i64
513   store double 0.000000e+00, double* %factual303
514   store i64 %ptwtointforfpstorage304, i64* %pointercast301
515   %intcast305 = ptrtoint i64* %fstorage274 to i64
516   %pmath306 = add i64 %intcast305, 40
517   %pointercast307 = inttoptr i64 %pmath306 to i64*
518   %malloccall308 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
519   %factual309 = bitcast i8* %malloccall308 to double*
520   %ptwtointforfpstorage310 = ptrtoint double* %factual309 to i64
521   store double 0.000000e+00, double* %factual309
522   store i64 %ptwtointforfpstorage310, i64* %pointercast307
523   %ptrtointfornlstorage311 = ptrtoint i64* %fstorage274 to i64
524   store i64 %ptrtointfornlstorage311, i64* %pointercast271
525   %ptwtointfornlstorage312 = ptrtoint i64* %pointercast271 to i64
526   %nextptr313 = add i64 %ptwtointfornlstorage312, 8
527   %nextptrptr314 = inttoptr i64 %nextptr313 to i64*
528   store i64* %immediate_tmp265, i64** %D0
529   %printf_ignore315 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([40 x i8], [40 x
          i8]* @"given survival rates and birth rates: \0A", i32 0, i32 0))
530   %L316 = load i64*, i64** %L
531   %_printf_prefix_2 = alloca i64
532   %_printf_prefix_1 = alloca i64
533   %intcast317 = ptrtoint i64* %L316 to i64
534   %pmath318 = add i64 %intcast317, 8
535   %pointercast319 = inttoptr i64 %pmath318 to i64*
536   store i64 0, i64* %_printf_prefix_2
537   br label %while
538
539 while:                                            ; preds = %merge, %entry
540   %_printf_prefix_2350 = load i64, i64* %_printf_prefix_2
541   %_slicesize_0 = load i64, i64* %pointercast319
542   %ftmp351 = sitofp i64 %_slicesize_0 to double
543   %ftmp352 = sitofp i64 %_printf_prefix_2350 to double
544   %tmp353 = fcmp olt double %ftmp352, %ftmp351
545   %booltmp354 = icmp ne i1 %tmp353, false
546   br i1 %booltmp354, label %while_body, label %merge355
547
548 while_body:                                       ; preds = %while
549   %gl = load i64, i64* %_printf_prefix_2
550   %intcast320 = ptrtoint i64* %L316 to i64
551   %pmath321 = add i64 %intcast320, 8
552   %pointercast322 = inttoptr i64 %pmath321 to i64*
```

```
553   %intcast323 = ptrtoint i64* %L316 to i64
554   %pmath324 = add i64 %intcast323, 16
555   %pointercast325 = inttoptr i64 %pmath324 to i64*
556   store i64 0, i64* %_printf_prefix_1
557   br label %while326
558 
559 while326:                                         ; preds = %while_body327, %while_body
560   %_printf_prefix_1344 = load i64, i64* %_printf_prefix_1
561   %_slicesize_1 = load i64, i64* %pointercast325
562   %ftmp = sitofp i64 %_slicesize_1 to double
563   %ftmp345 = sitofp i64 %_printf_prefix_1344 to double
564   %tmp346 = fcmp olt double %ftmp345, %ftmp
565   %booltmp = icmp ne i1 %tmp346, false
566   br i1 %booltmp, label %while_body327, label %merge
567 
568 while_body327:                                    ; preds = %while326
569   %gl328 = load i64, i64* %_printf_prefix_1
570   %gl329 = load i64, i64* %_printf_prefix_2
571   %offsetload = load i64, i64* %L316
572   %plus1 = add i64 %offsetload, 1
573   %intcast330 = ptrtoint i64* %L316 to i64
574   %offsetcalc = mul i64 %plus1, 8
575   %pmath331 = add i64 %intcast330, %offsetcalc
576   %pointercast332 = inttoptr i64 %pmath331 to i64*
577   %rootload = load i64, i64* %pointercast332
578   %offsetcalc333 = mul i64 %gl329, 8
579   %pmath334 = add i64 %rootload, %offsetcalc333
580   %pointercast335 = inttoptr i64 %pmath334 to i64*
581   %deref = load i64, i64* %pointercast335
582   %derefpntr = inttoptr i64 %deref to i64*
583   %intcast336 = ptrtoint i64* %derefpntr to i64
584   %offsetcalc337 = mul i64 %gl328, 8
585   %pmath338 = add i64 %intcast336, %offsetcalc337
586   %pointercast339 = inttoptr i64 %pmath338 to i64*
587   %deref340 = load i64, i64* %pointercast339
588   %derefpntr341 = inttoptr i64 %deref340 to i64*
589   %fpaccess = bitcast i64* %derefpntr341 to double*
590   %fpret = load double, double* %fpaccess
591   %printf_ignore342 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([7 x i8], [7 x i8]*
          @fmt3, i32 0, i32 0), double %fpret)
592   %_printf_prefix_1343 = load i64, i64* %_printf_prefix_1
593   %tmp = add i64 1, %_printf_prefix_1343
594   store i64 %tmp, i64* %_printf_prefix_1
595   br label %while326
596 
597 merge:                                            ; preds = %while326
598   %printf_ignore347 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([2 x i8], [2 x i8]*
          @nlstr, i32 0, i32 0))
599   %_printf_prefix_2348 = load i64, i64* %_printf_prefix_2
600   %tmp349 = add i64 1, %_printf_prefix_2348
601   store i64 %tmp349, i64* %_printf_prefix_2
602   br label %while
603 
604 merge355:                                         ; preds = %while
605   %printf_ignore356 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([2 x i8], [2 x i8]*
          @nlstr, i32 0, i32 0))
```

```
606   %printf_ignore357 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([48 x i8], [48 x
          i8]* @"you want to know what your age distribution is\0A", i32 0, i32 0))
607   %printf_ignore358 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([16 x i8], [16 x
          i8]* @"after 2 years: ", i32 0, i32 0))
608   %D = alloca i64*
609   %free_alloc = alloca i64
610   store i64 0, i64* %free_alloc
611   %free_alloc359 = alloca i64
612   store i64 0, i64* %free_alloc359
613   %L360 = load i64*, i64** %L
614   %free_alloc361 = alloca i64
615   store i64 0, i64* %free_alloc361
616   %D0362 = load i64*, i64** %D0
617   %intcast363 = ptrtoint i64* %L360 to i64
618   %pmath364 = add i64 %intcast363, 8
619   %pointercast365 = inttoptr i64 %pmath364 to i64*
620   %intcast366 = ptrtoint i64* %L360 to i64
621   %pmath367 = add i64 %intcast366, 16
622   %pointercast368 = inttoptr i64 %pmath367 to i64*
623   %10 = trunc i64 3 to i32
624   %mallocsize369 = mul i32 %10, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
625   %malloccall370 = tail call i8* @malloc(i32 %mallocsize369)
626   %static_ten = bitcast i8* %malloccall370 to i64*
627   store i64 1, i64* %static_ten
628   %hdrint = ptrtoint i64* %L360 to i64
629   %hdrint2 = ptrtoint i64* %D0362 to i64
630   %hdrcpycalc = add i64 %hdrint, 8
631   %dstint = ptrtoint i64* %static_ten to i64
632   %tintoffset = add i64 %dstint, 8
633   %srcptr = inttoptr i64 %hdrcpycalc to i64*
634   %dstptr = inttoptr i64 %tintoffset to i64*
635   %memload = load i64, i64* %srcptr
636   store i64 %memload, i64* %dstptr
637   %tgt_header = alloca i64*
638   %s1_header = alloca i64*
639   %s2_header = alloca i64*
640   store i64* %static_ten, i64** %tgt_header
641   store i64* %L360, i64** %s1_header
642   store i64* %D0362, i64** %s2_header
643   %a = alloca i64
644   %b = alloca i64
645   %intcast371 = ptrtoint i64* %static_ten to i64
646   %pmath372 = add i64 %intcast371, 8
647   %pointercast373 = inttoptr i64 %pmath372 to i64*
648   %slicesizeload = load i64, i64* %pointercast373
649   %offsetload374 = load i64, i64* %static_ten
650   %plus1375 = add i64 %offsetload374, 1
651   %intcast376 = ptrtoint i64* %static_ten to i64
652   %offsetcalc377 = mul i64 %plus1375, 8
653   %pmath378 = add i64 %intcast376, %offsetcalc377
654   %pointercast379 = inttoptr i64 %pmath378 to i64*
655   %11 = trunc i64 %slicesizeload to i32
656   %mallocsize380 = mul i32 %11, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
657   %malloccall381 = tail call i8* @malloc(i32 %mallocsize380)
658   %mcarr = bitcast i8* %malloccall381 to i64*
659   %mcarrptrtoint = ptrtoint i64* %mcarr to i64
```

```
660   store i64 %mcarrptrtoint, i64* %pointercast379
661   store i64 0, i64* %a
662   br label %while382
663
664 while382:                                        ; preds = %merge500, %merge355
665   %a503 = load i64, i64* %a
666   %_slicesize_0504 = load i64, i64* %pointercast373
667   %ftmp505 = sitofp i64 %_slicesize_0504 to double
668   %ftmp506 = sitofp i64 %a503 to double
669   %tmp507 = fcmp olt double %ftmp506, %ftmp505
670   %booltmp508 = icmp ne i1 %tmp507, false
671   br i1 %booltmp508, label %while_body383, label %merge509
672
673 while_body383:                                    ; preds = %while382
674   %gl384 = load i64, i64* %a
675   %offsetload385 = load i64, i64* %static_ten
676   %plus1386 = add i64 %offsetload385, 1
677   %intcast387 = ptrtoint i64* %static_ten to i64
678   %offsetcalc388 = mul i64 %plus1386, 8
679   %pmath389 = add i64 %intcast387, %offsetcalc388
680   %pointercast390 = inttoptr i64 %pmath389 to i64*
681   %deref391 = load i64, i64* %pointercast390
682   %offsetcalc392 = mul i64 %gl384, 8
683   %pmath393 = add i64 %deref391, %offsetcalc392
684   %pointercast394 = inttoptr i64 %pmath393 to i64*
685   %malloccall395 = tail call i8* @malloc(i32 ptrtoint (i64* getelementptr (i64, i64* null, i32 1)
          to i32))
686   %abldrk = bitcast i8* %malloccall395 to i64*
687   %abldrkcast = ptrtoint i64* %abldrk to i64
688   store i64 %abldrkcast, i64* %pointercast394
689   %a396 = load i64, i64* %a
690   %free_alloc397 = alloca i64
691   store i64 %a396, i64* %free_alloc397
692   %_tgt_header_ = load i64*, i64** %tgt_header
693   %glr = load i64, i64* %free_alloc397
694   %offsetload398 = load i64, i64* %_tgt_header_
695   %plus1399 = add i64 %offsetload398, 1
696   %intcast400 = ptrtoint i64* %_tgt_header_ to i64
697   %offsetcalc401 = mul i64 %plus1399, 8
698   %pmath402 = add i64 %intcast400, %offsetcalc401
699   %pointercast403 = inttoptr i64 %pmath402 to i64*
700   %rootload404 = load i64, i64* %pointercast403
701   %offsetcalc405 = mul i64 %glr, 8
702   %pmath406 = add i64 %rootload404, %offsetcalc405
703   %pointercast407 = inttoptr i64 %pmath406 to i64*
704   %deref408 = load i64, i64* %pointercast407
705   %derefpntr409 = inttoptr i64 %deref408 to i64*
706   %fpaccess410 = bitcast i64* %derefpntr409 to double*
707   store double 0.000000e+00, double* %fpaccess410
708   store i64 0, i64* %b
709   br label %while411
710
711 while411:                                        ; preds = %while_body412, %while_body383
712   %b495 = load i64, i64* %b
713   %_tmult_size_inner_0 = load i64, i64* %pointercast368
714   %ftmp496 = sitofp i64 %_tmult_size_inner_0 to double
```

```
  %ftmp497 = sitofp i64 %b495 to double
  %tmp498 = fcmp olt double %ftmp497, %ftmp496
  %booltmp499 = icmp ne i1 %tmp498, false
  br i1 %booltmp499, label %while_body412, label %merge500

while_body412:                                    ; preds = %while411
  %a413 = load i64, i64* %a
  %free_alloc414 = alloca i64
  store i64 %a413, i64* %free_alloc414
  %_tgt_header_415 = load i64*, i64** %tgt_header
  %glr416 = load i64, i64* %free_alloc414
  %offsetload417 = load i64, i64* %_tgt_header_415
  %plus1418 = add i64 %offsetload417, 1
  %intcast419 = ptrtoint i64* %_tgt_header_415 to i64
  %offsetcalc420 = mul i64 %plus1418, 8
  %pmath421 = add i64 %intcast419, %offsetcalc420
  %pointercast422 = inttoptr i64 %pmath421 to i64*
  %rootload423 = load i64, i64* %pointercast422
  %offsetcalc424 = mul i64 %glr416, 8
  %pmath425 = add i64 %rootload423, %offsetcalc424
  %pointercast426 = inttoptr i64 %pmath425 to i64*
  %deref427 = load i64, i64* %pointercast426
  %derefpntr428 = inttoptr i64 %deref427 to i64*
  %fpaccess429 = bitcast i64* %derefpntr428 to double*
  %fpret430 = load double, double* %fpaccess429
  %a431 = load i64, i64* %a
  %free_alloc432 = alloca i64
  store i64 %a431, i64* %free_alloc432
  %b433 = load i64, i64* %b
  %free_alloc434 = alloca i64
  store i64 %b433, i64* %free_alloc434
  %_s1_header_ = load i64*, i64** %s1_header
  %glr435 = load i64, i64* %free_alloc434
  %glr436 = load i64, i64* %free_alloc432
  %offsetload437 = load i64, i64* %_s1_header_
  %plus1438 = add i64 %offsetload437, 1
  %intcast439 = ptrtoint i64* %_s1_header_ to i64
  %offsetcalc440 = mul i64 %plus1438, 8
  %pmath441 = add i64 %intcast439, %offsetcalc440
  %pointercast442 = inttoptr i64 %pmath441 to i64*
  %rootload443 = load i64, i64* %pointercast442
  %offsetcalc444 = mul i64 %glr436, 8
  %pmath445 = add i64 %rootload443, %offsetcalc444
  %pointercast446 = inttoptr i64 %pmath445 to i64*
  %deref447 = load i64, i64* %pointercast446
  %derefpntr448 = inttoptr i64 %deref447 to i64*
  %intcast449 = ptrtoint i64* %derefpntr448 to i64
  %offsetcalc450 = mul i64 %glr435, 8
  %pmath451 = add i64 %intcast449, %offsetcalc450
  %pointercast452 = inttoptr i64 %pmath451 to i64*
  %deref453 = load i64, i64* %pointercast452
  %derefpntr454 = inttoptr i64 %deref453 to i64*
  %fpaccess455 = bitcast i64* %derefpntr454 to double*
  %fpret456 = load double, double* %fpaccess455
  %b457 = load i64, i64* %b
  %free_alloc458 = alloca i64
```

```
771   store i64 %b457, i64* %free_alloc458
772   %_s2_header_ = load i64*, i64** %s2_header
773   %glr459 = load i64, i64* %free_alloc458
774   %offsetload460 = load i64, i64* %_s2_header_
775   %plus1461 = add i64 %offsetload460, 1
776   %intcast462 = ptrtoint i64* %_s2_header_ to i64
777   %offsetcalc463 = mul i64 %plus1461, 8
778   %pmath464 = add i64 %intcast462, %offsetcalc463
779   %pointercast465 = inttoptr i64 %pmath464 to i64*
780   %rootload466 = load i64, i64* %pointercast465
781   %offsetcalc467 = mul i64 %glr459, 8
782   %pmath468 = add i64 %rootload466, %offsetcalc467
783   %pointercast469 = inttoptr i64 %pmath468 to i64*
784   %deref470 = load i64, i64* %pointercast469
785   %derefpntr471 = inttoptr i64 %deref470 to i64*
786   %fpaccess472 = bitcast i64* %derefpntr471 to double*
787   %fpret473 = load double, double* %fpaccess472
788   %tmp474 = fmul double %fpret456, %fpret473
789   %tmp475 = fadd double %fpret430, %tmp474
790   %a476 = load i64, i64* %a
791   %free_alloc477 = alloca i64
792   store i64 %a476, i64* %free_alloc477
793   %_tgt_header_478 = load i64*, i64** %tgt_header
794   %glr479 = load i64, i64* %free_alloc477
795   %offsetload480 = load i64, i64* %_tgt_header_478
796   %plus1481 = add i64 %offsetload480, 1
797   %intcast482 = ptrtoint i64* %_tgt_header_478 to i64
798   %offsetcalc483 = mul i64 %plus1481, 8
799   %pmath484 = add i64 %intcast482, %offsetcalc483
800   %pointercast485 = inttoptr i64 %pmath484 to i64*
801   %rootload486 = load i64, i64* %pointercast485
802   %offsetcalc487 = mul i64 %glr479, 8
803   %pmath488 = add i64 %rootload486, %offsetcalc487
804   %pointercast489 = inttoptr i64 %pmath488 to i64*
805   %deref490 = load i64, i64* %pointercast489
806   %derefpntr491 = inttoptr i64 %deref490 to i64*
807   %fpaccess492 = bitcast i64* %derefpntr491 to double*
808   store double %tmp475, double* %fpaccess492
809   %b493 = load i64, i64* %b
810   %tmp494 = add i64 1, %b493
811   store i64 %tmp494, i64* %b
812   br label %while411
813
814 merge500:                                         ; preds = %while411
815   %a501 = load i64, i64* %a
816   %tmp502 = add i64 1, %a501
817   store i64 %tmp502, i64* %a
818   br label %while382
819
820 merge509:                                         ; preds = %while382
821   %hdraddr_load = load i64*, i64** %tgt_header
822   store i64* %hdraddr_load, i64** %D
823   %D510 = load i64*, i64** %D
824   %_printf_prefix_1511 = alloca i64
825   %intcast512 = ptrtoint i64* %D510 to i64
826   %pmath513 = add i64 %intcast512, 8
```

```
827   %pointercast514 = inttoptr i64 %pmath513 to i64*
828   store i64 0, i64* %_printf_prefix_1511
829   br label %while515
830 
831 while515:                                         ; preds = %while_body516, %merge509
832   %_printf_prefix_1535 = load i64, i64* %_printf_prefix_1511
833   %_slicesize_0536 = load i64, i64* %pointercast514
834   %ftmp537 = sitofp i64 %_slicesize_0536 to double
835   %ftmp538 = sitofp i64 %_printf_prefix_1535 to double
836   %tmp539 = fcmp olt double %ftmp538, %ftmp537
837   %booltmp540 = icmp ne i1 %tmp539, false
838   br i1 %booltmp540, label %while_body516, label %merge541
839 
840 while_body516:                                    ; preds = %while515
841   %gl517 = load i64, i64* %_printf_prefix_1511
842   %offsetload518 = load i64, i64* %D510
843   %plus1519 = add i64 %offsetload518, 1
844   %intcast520 = ptrtoint i64* %D510 to i64
845   %offsetcalc521 = mul i64 %plus1519, 8
846   %pmath522 = add i64 %intcast520, %offsetcalc521
847   %pointercast523 = inttoptr i64 %pmath522 to i64*
848   %rootload524 = load i64, i64* %pointercast523
849   %offsetcalc525 = mul i64 %gl517, 8
850   %pmath526 = add i64 %rootload524, %offsetcalc525
851   %pointercast527 = inttoptr i64 %pmath526 to i64*
852   %deref528 = load i64, i64* %pointercast527
853   %derefpntr529 = inttoptr i64 %deref528 to i64*
854   %fpaccess530 = bitcast i64* %derefpntr529 to double*
855   %fpret531 = load double, double* %fpaccess530
856   %printf_ignore532 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([7 x i8], [7 x i8]*
          @fmt3, i32 0, i32 0), double %fpret531)
857   %_printf_prefix_1533 = load i64, i64* %_printf_prefix_1511
858   %tmp534 = add i64 1, %_printf_prefix_1533
859   store i64 %tmp534, i64* %_printf_prefix_1511
860   br label %while515
861 
862 merge541:                                         ; preds = %while515
863   %printf_ignore542 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([2 x i8], [2 x i8]*
          @nlstr, i32 0, i32 0))
864   %i = alloca i64
865   store i64 0, i64* %i
866   br label %while543
867 
868 while543:                                         ; preds = %merge720, %merge541
869   %i724 = load i64, i64* %i
870   %ftmp725 = sitofp i64 %i724 to double
871   %tmp726 = fcmp olt double %ftmp725, 1.400000e+01
872   %booltmp727 = icmp ne i1 %tmp726, false
873   br i1 %booltmp727, label %while_body544, label %merge728
874 
875 while_body544:                                    ; preds = %while543
876   %free_alloc545 = alloca i64
877   store i64 0, i64* %free_alloc545
878   %free_alloc546 = alloca i64
879   store i64 0, i64* %free_alloc546
880   %L547 = load i64*, i64** %L
```

```
  %free_alloc548 = alloca i64
  store i64 0, i64* %free_alloc548
  %D549 = load i64*, i64** %D
  %intcast550 = ptrtoint i64* %L547 to i64
  %pmath551 = add i64 %intcast550, 8
  %pointercast552 = inttoptr i64 %pmath551 to i64*
  %intcast553 = ptrtoint i64* %L547 to i64
  %pmath554 = add i64 %intcast553, 16
  %pointercast555 = inttoptr i64 %pmath554 to i64*
  %12 = trunc i64 3 to i32
  %mallocsize556 = mul i32 %12, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall557 = tail call i8* @malloc(i32 %mallocsize556)
  %static_ten558 = bitcast i8* %malloccall557 to i64*
  store i64 1, i64* %static_ten558
  %hdrint559 = ptrtoint i64* %L547 to i64
  %hdrint2560 = ptrtoint i64* %D549 to i64
  %hdrcpycalc561 = add i64 %hdrint559, 8
  %dstint562 = ptrtoint i64* %static_ten558 to i64
  %tintoffset563 = add i64 %dstint562, 8
  %srcptr564 = inttoptr i64 %hdrcpycalc561 to i64*
  %dstptr565 = inttoptr i64 %tintoffset563 to i64*
  %memload566 = load i64, i64* %srcptr564
  store i64 %memload566, i64* %dstptr565
  %tgt_header567 = alloca i64*
  %s1_header568 = alloca i64*
  %s2_header569 = alloca i64*
  store i64* %static_ten558, i64** %tgt_header567
  store i64* %L547, i64** %s1_header568
  store i64* %D549, i64** %s2_header569
  %a570 = alloca i64
  %b571 = alloca i64
  %intcast572 = ptrtoint i64* %static_ten558 to i64
  %pmath573 = add i64 %intcast572, 8
  %pointercast574 = inttoptr i64 %pmath573 to i64*
  %slicesizeload575 = load i64, i64* %pointercast574
  %offsetload576 = load i64, i64* %static_ten558
  %plus1577 = add i64 %offsetload576, 1
  %intcast578 = ptrtoint i64* %static_ten558 to i64
  %offsetcalc579 = mul i64 %plus1577, 8
  %pmath580 = add i64 %intcast578, %offsetcalc579
  %pointercast581 = inttoptr i64 %pmath580 to i64*
  %13 = trunc i64 %slicesizeload575 to i32
  %mallocsize582 = mul i32 %13, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall583 = tail call i8* @malloc(i32 %mallocsize582)
  %mcarr584 = bitcast i8* %malloccall583 to i64*
  %mcarrptrtoint585 = ptrtoint i64* %mcarr584 to i64
  store i64 %mcarrptrtoint585, i64* %pointercast581
  store i64 0, i64* %a570
  br label %while586

while586:                                         ; preds = %merge711, %while_body544
  %a714 = load i64, i64* %a570
  %_slicesize_0715 = load i64, i64* %pointercast574
  %ftmp716 = sitofp i64 %_slicesize_0715 to double
  %ftmp717 = sitofp i64 %a714 to double
  %tmp718 = fcmp olt double %ftmp717, %ftmp716
```

```
  %booltmp719 = icmp ne i1 %tmp718, false
  br i1 %booltmp719, label %while_body587, label %merge720

while_body587:                                    ; preds = %while586
  %gl588 = load i64, i64* %a570
  %offsetload589 = load i64, i64* %static_ten558
  %plus1590 = add i64 %offsetload589, 1
  %intcast591 = ptrtoint i64* %static_ten558 to i64
  %offsetcalc592 = mul i64 %plus1590, 8
  %pmath593 = add i64 %intcast591, %offsetcalc592
  %pointercast594 = inttoptr i64 %pmath593 to i64*
  %deref595 = load i64, i64* %pointercast594
  %offsetcalc596 = mul i64 %gl588, 8
  %pmath597 = add i64 %deref595, %offsetcalc596
  %pointercast598 = inttoptr i64 %pmath597 to i64*
  %malloccall599 = tail call i8* @malloc(i32 ptrtoint (i64* getelementptr (i64, i64* null, i32 1)
      to i32))
  %abldrk600 = bitcast i8* %malloccall599 to i64*
  %abldrkcast601 = ptrtoint i64* %abldrk600 to i64
  store i64 %abldrkcast601, i64* %pointercast598
  %a602 = load i64, i64* %a570
  %free_alloc603 = alloca i64
  store i64 %a602, i64* %free_alloc603
  %_tgt_header_604 = load i64*, i64** %tgt_header567
  %glr605 = load i64, i64* %free_alloc603
  %offsetload606 = load i64, i64* %_tgt_header_604
  %plus1607 = add i64 %offsetload606, 1
  %intcast608 = ptrtoint i64* %_tgt_header_604 to i64
  %offsetcalc609 = mul i64 %plus1607, 8
  %pmath610 = add i64 %intcast608, %offsetcalc609
  %pointercast611 = inttoptr i64 %pmath610 to i64*
  %rootload612 = load i64, i64* %pointercast611
  %offsetcalc613 = mul i64 %glr605, 8
  %pmath614 = add i64 %rootload612, %offsetcalc613
  %pointercast615 = inttoptr i64 %pmath614 to i64*
  %deref616 = load i64, i64* %pointercast615
  %derefpntr617 = inttoptr i64 %deref616 to i64*
  %fpaccess618 = bitcast i64* %derefpntr617 to double*
  store double 0.000000e+00, double* %fpaccess618
  store i64 0, i64* %b571
  br label %while619

while619:                                         ; preds = %while_body620, %while_body587
  %b705 = load i64, i64* %b571
  %_tmult_size_inner_0706 = load i64, i64* %pointercast555
  %ftmp707 = sitofp i64 %_tmult_size_inner_0706 to double
  %ftmp708 = sitofp i64 %b705 to double
  %tmp709 = fcmp olt double %ftmp708, %ftmp707
  %booltmp710 = icmp ne i1 %tmp709, false
  br i1 %booltmp710, label %while_body620, label %merge711

while_body620:                                    ; preds = %while619
  %a621 = load i64, i64* %a570
  %free_alloc622 = alloca i64
  store i64 %a621, i64* %free_alloc622
  %_tgt_header_623 = load i64*, i64** %tgt_header567
```

```
  %glr624 = load i64, i64* %free_alloc622
  %offsetload625 = load i64, i64* %_tgt_header_623
  %plus1626 = add i64 %offsetload625, 1
  %intcast627 = ptrtoint i64* %_tgt_header_623 to i64
  %offsetcalc628 = mul i64 %plus1626, 8
  %pmath629 = add i64 %intcast627, %offsetcalc628
  %pointercast630 = inttoptr i64 %pmath629 to i64*
  %rootload631 = load i64, i64* %pointercast630
  %offsetcalc632 = mul i64 %glr624, 8
  %pmath633 = add i64 %rootload631, %offsetcalc632
  %pointercast634 = inttoptr i64 %pmath633 to i64*
  %deref635 = load i64, i64* %pointercast634
  %derefpntr636 = inttoptr i64 %deref635 to i64*
  %fpaccess637 = bitcast i64* %derefpntr636 to double*
  %fpret638 = load double, double* %fpaccess637
  %a639 = load i64, i64* %a570
  %free_alloc640 = alloca i64
  store i64 %a639, i64* %free_alloc640
  %b641 = load i64, i64* %b571
  %free_alloc642 = alloca i64
  store i64 %b641, i64* %free_alloc642
  %_s1_header_643 = load i64*, i64** %s1_header568
  %glr644 = load i64, i64* %free_alloc642
  %glr645 = load i64, i64* %free_alloc640
  %offsetload646 = load i64, i64* %_s1_header_643
  %plus1647 = add i64 %offsetload646, 1
  %intcast648 = ptrtoint i64* %_s1_header_643 to i64
  %offsetcalc649 = mul i64 %plus1647, 8
  %pmath650 = add i64 %intcast648, %offsetcalc649
  %pointercast651 = inttoptr i64 %pmath650 to i64*
  %rootload652 = load i64, i64* %pointercast651
  %offsetcalc653 = mul i64 %glr645, 8
  %pmath654 = add i64 %rootload652, %offsetcalc653
  %pointercast655 = inttoptr i64 %pmath654 to i64*
  %deref656 = load i64, i64* %pointercast655
  %derefpntr657 = inttoptr i64 %deref656 to i64*
  %intcast658 = ptrtoint i64* %derefpntr657 to i64
  %offsetcalc659 = mul i64 %glr644, 8
  %pmath660 = add i64 %intcast658, %offsetcalc659
  %pointercast661 = inttoptr i64 %pmath660 to i64*
  %deref662 = load i64, i64* %pointercast661
  %derefpntr663 = inttoptr i64 %deref662 to i64*
  %fpaccess664 = bitcast i64* %derefpntr663 to double*
  %fpret665 = load double, double* %fpaccess664
  %b666 = load i64, i64* %b571
  %free_alloc667 = alloca i64
  store i64 %b666, i64* %free_alloc667
  %_s2_header_668 = load i64*, i64** %s2_header569
  %glr669 = load i64, i64* %free_alloc667
  %offsetload670 = load i64, i64* %_s2_header_668
  %plus1671 = add i64 %offsetload670, 1
  %intcast672 = ptrtoint i64* %_s2_header_668 to i64
  %offsetcalc673 = mul i64 %plus1671, 8
  %pmath674 = add i64 %intcast672, %offsetcalc673
  %pointercast675 = inttoptr i64 %pmath674 to i64*
  %rootload676 = load i64, i64* %pointercast675
```

```
  %offsetcalc677 = mul i64 %glr669, 8
  %pmath678 = add i64 %rootload676, %offsetcalc677
  %pointercast679 = inttoptr i64 %pmath678 to i64*
  %deref680 = load i64, i64* %pointercast679
  %derefpntr681 = inttoptr i64 %deref680 to i64*
  %fpaccess682 = bitcast i64* %derefpntr681 to double*
  %fpret683 = load double, double* %fpaccess682
  %tmp684 = fmul double %fpret665, %fpret683
  %tmp685 = fadd double %fpret638, %tmp684
  %a686 = load i64, i64* %a570
  %free_alloc687 = alloca i64
  store i64 %a686, i64* %free_alloc687
  %_tgt_header_688 = load i64*, i64** %tgt_header567
  %glr689 = load i64, i64* %free_alloc687
  %offsetload690 = load i64, i64* %_tgt_header_688
  %plus1691 = add i64 %offsetload690, 1
  %intcast692 = ptrtoint i64* %_tgt_header_688 to i64
  %offsetcalc693 = mul i64 %plus1691, 8
  %pmath694 = add i64 %intcast692, %offsetcalc693
  %pointercast695 = inttoptr i64 %pmath694 to i64*
  %rootload696 = load i64, i64* %pointercast695
  %offsetcalc697 = mul i64 %glr689, 8
  %pmath698 = add i64 %rootload696, %offsetcalc697
  %pointercast699 = inttoptr i64 %pmath698 to i64*
  %deref700 = load i64, i64* %pointercast699
  %derefpntr701 = inttoptr i64 %deref700 to i64*
  %fpaccess702 = bitcast i64* %derefpntr701 to double*
  store double %tmp685, double* %fpaccess702
  %b703 = load i64, i64* %b571
  %tmp704 = add i64 1, %b703
  store i64 %tmp704, i64* %b571
  br label %while619

merge711:                                         ; preds = %while619
  %a712 = load i64, i64* %a570
  %tmp713 = add i64 1, %a712
  store i64 %tmp713, i64* %a570
  br label %while586

merge720:                                         ; preds = %while586
  %hdraddr_load721 = load i64*, i64** %tgt_header567
  store i64* %hdraddr_load721, i64** %D
  %i722 = load i64, i64* %i
  %tmp723 = add i64 %i722, 1
  store i64 %tmp723, i64* %i
  br label %while543

merge728:                                         ; preds = %while543
  %printf_ignore729 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([17 x i8], [17 x
      i8]* @"after 30 years: ", i32 0, i32 0))
  %D730 = load i64*, i64** %D
  %_printf_prefix_1731 = alloca i64
  %intcast732 = ptrtoint i64* %D730 to i64
  %pmath733 = add i64 %intcast732, 8
  %pointercast734 = inttoptr i64 %pmath733 to i64*
  store i64 0, i64* %_printf_prefix_1731
```

```
1103   br label %while735
1104
1105 while735:                                          ; preds = %while_body736, %merge728
1106   %_printf_prefix_1755 = load i64, i64* %_printf_prefix_1731
1107   %_slicesize_0756 = load i64, i64* %pointercast734
1108   %ftmp757 = sitofp i64 %_slicesize_0756 to double
1109   %ftmp758 = sitofp i64 %_printf_prefix_1755 to double
1110   %tmp759 = fcmp olt double %ftmp758, %ftmp757
1111   %booltmp760 = icmp ne i1 %tmp759, false
1112   br i1 %booltmp760, label %while_body736, label %merge761
1113
1114 while_body736:                                     ; preds = %while735
1115   %gl737 = load i64, i64* %_printf_prefix_1731
1116   %offsetload738 = load i64, i64* %D730
1117   %plus1739 = add i64 %offsetload738, 1
1118   %intcast740 = ptrtoint i64* %D730 to i64
1119   %offsetcalc741 = mul i64 %plus1739, 8
1120   %pmath742 = add i64 %intcast740, %offsetcalc741
1121   %pointercast743 = inttoptr i64 %pmath742 to i64*
1122   %rootload744 = load i64, i64* %pointercast743
1123   %offsetcalc745 = mul i64 %gl737, 8
1124   %pmath746 = add i64 %rootload744, %offsetcalc745
1125   %pointercast747 = inttoptr i64 %pmath746 to i64*
1126   %deref748 = load i64, i64* %pointercast747
1127   %derefpntr749 = inttoptr i64 %deref748 to i64*
1128   %fpaccess750 = bitcast i64* %derefpntr749 to double*
1129   %fpret751 = load double, double* %fpaccess750
1130   %printf_ignore752 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([7 x i8], [7 x i8]*
          @fmt3, i32 0, i32 0), double %fpret751)
1131   %_printf_prefix_1753 = load i64, i64* %_printf_prefix_1731
1132   %tmp754 = add i64 1, %_printf_prefix_1753
1133   store i64 %tmp754, i64* %_printf_prefix_1731
1134   br label %while735
1135
1136 merge761:                                          ; preds = %while735
1137   %printf_ignore762 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([2 x i8], [2 x i8]*
          @nlstr, i32 0, i32 0))
1138   %i763 = alloca i64
1139   store i64 0, i64* %i763
1140   br label %while764
1141
1142 while764:                                          ; preds = %merge941, %merge761
1143   %i945 = load i64, i64* %i763
1144   %ftmp946 = sitofp i64 %i945 to double
1145   %tmp947 = fcmp olt double %ftmp946, 2.600000e+01
1146   %booltmp948 = icmp ne i1 %tmp947, false
1147   br i1 %booltmp948, label %while_body765, label %merge949
1148
1149 while_body765:                                     ; preds = %while764
1150   %free_alloc766 = alloca i64
1151   store i64 0, i64* %free_alloc766
1152   %free_alloc767 = alloca i64
1153   store i64 0, i64* %free_alloc767
1154   %L768 = load i64*, i64** %L
1155   %free_alloc769 = alloca i64
1156   store i64 0, i64* %free_alloc769
```

```
  %D770 = load i64*, i64** %D
  %intcast771 = ptrtoint i64* %L768 to i64
  %pmath772 = add i64 %intcast771, 8
  %pointercast773 = inttoptr i64 %pmath772 to i64*
  %intcast774 = ptrtoint i64* %L768 to i64
  %pmath775 = add i64 %intcast774, 16
  %pointercast776 = inttoptr i64 %pmath775 to i64*
  %14 = trunc i64 3 to i32
  %mallocsize777 = mul i32 %14, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall778 = tail call i8* @malloc(i32 %mallocsize777)
  %static_ten779 = bitcast i8* %malloccall778 to i64*
  store i64 1, i64* %static_ten779
  %hdrint780 = ptrtoint i64* %L768 to i64
  %hdrint2781 = ptrtoint i64* %D770 to i64
  %hdrcpycalc782 = add i64 %hdrint780, 8
  %dstint783 = ptrtoint i64* %static_ten779 to i64
  %tintoffset784 = add i64 %dstint783, 8
  %srcptr785 = inttoptr i64 %hdrcpycalc782 to i64*
  %dstptr786 = inttoptr i64 %tintoffset784 to i64*
  %memload787 = load i64, i64* %srcptr785
  store i64 %memload787, i64* %dstptr786
  %tgt_header788 = alloca i64*
  %s1_header789 = alloca i64*
  %s2_header790 = alloca i64*
  store i64* %static_ten779, i64** %tgt_header788
  store i64* %L768, i64** %s1_header789
  store i64* %D770, i64** %s2_header790
  %a791 = alloca i64
  %b792 = alloca i64
  %intcast793 = ptrtoint i64* %static_ten779 to i64
  %pmath794 = add i64 %intcast793, 8
  %pointercast795 = inttoptr i64 %pmath794 to i64*
  %slicesizeload796 = load i64, i64* %pointercast795
  %offsetload797 = load i64, i64* %static_ten779
  %plus1798 = add i64 %offsetload797, 1
  %intcast799 = ptrtoint i64* %static_ten779 to i64
  %offsetcalc800 = mul i64 %plus1798, 8
  %pmath801 = add i64 %intcast799, %offsetcalc800
  %pointercast802 = inttoptr i64 %pmath801 to i64*
  %15 = trunc i64 %slicesizeload796 to i32
  %mallocsize803 = mul i32 %15, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall804 = tail call i8* @malloc(i32 %mallocsize803)
  %mcarr805 = bitcast i8* %malloccall804 to i64*
  %mcarrptrtoint806 = ptrtoint i64* %mcarr805 to i64
  store i64 %mcarrptrtoint806, i64* %pointercast802
  store i64 0, i64* %a791
  br label %while807

while807:                                         ; preds = %merge932, %while_body765
  %a935 = load i64, i64* %a791
  %_slicesize_0936 = load i64, i64* %pointercast795
  %ftmp937 = sitofp i64 %_slicesize_0936 to double
  %ftmp938 = sitofp i64 %a935 to double
  %tmp939 = fcmp olt double %ftmp938, %ftmp937
  %booltmp940 = icmp ne i1 %tmp939, false
  br i1 %booltmp940, label %while_body808, label %merge941
```

```
1213
1214 while_body808:                                    ; preds = %while807
1215   %gl809 = load i64, i64* %a791
1216   %offsetload810 = load i64, i64* %static_ten779
1217   %plus1811 = add i64 %offsetload810, 1
1218   %intcast812 = ptrtoint i64* %static_ten779 to i64
1219   %offsetcalc813 = mul i64 %plus1811, 8
1220   %pmath814 = add i64 %intcast812, %offsetcalc813
1221   %pointercast815 = inttoptr i64 %pmath814 to i64*
1222   %deref816 = load i64, i64* %pointercast815
1223   %offsetcalc817 = mul i64 %gl809, 8
1224   %pmath818 = add i64 %deref816, %offsetcalc817
1225   %pointercast819 = inttoptr i64 %pmath818 to i64*
1226   %malloccall820 = tail call i8* @malloc(i32 ptrtoint (i64* getelementptr (i64, i64* null, i32 1)
          to i32))
1227   %abldrk821 = bitcast i8* %malloccall820 to i64*
1228   %abldrkcast822 = ptrtoint i64* %abldrk821 to i64
1229   store i64 %abldrkcast822, i64* %pointercast819
1230   %a823 = load i64, i64* %a791
1231   %free_alloc824 = alloca i64
1232   store i64 %a823, i64* %free_alloc824
1233   %_tgt_header_825 = load i64*, i64** %tgt_header788
1234   %glr826 = load i64, i64* %free_alloc824
1235   %offsetload827 = load i64, i64* %_tgt_header_825
1236   %plus1828 = add i64 %offsetload827, 1
1237   %intcast829 = ptrtoint i64* %_tgt_header_825 to i64
1238   %offsetcalc830 = mul i64 %plus1828, 8
1239   %pmath831 = add i64 %intcast829, %offsetcalc830
1240   %pointercast832 = inttoptr i64 %pmath831 to i64*
1241   %rootload833 = load i64, i64* %pointercast832
1242   %offsetcalc834 = mul i64 %glr826, 8
1243   %pmath835 = add i64 %rootload833, %offsetcalc834
1244   %pointercast836 = inttoptr i64 %pmath835 to i64*
1245   %deref837 = load i64, i64* %pointercast836
1246   %derefpntr838 = inttoptr i64 %deref837 to i64*
1247   %fpaccess839 = bitcast i64* %derefpntr838 to double*
1248   store double 0.000000e+00, double* %fpaccess839
1249   store i64 0, i64* %b792
1250   br label %while840
1251
1252 while840:                                         ; preds = %while_body841, %while_body808
1253   %b926 = load i64, i64* %b792
1254   %_tmult_size_inner_0927 = load i64, i64* %pointercast776
1255   %ftmp928 = sitofp i64 %_tmult_size_inner_0927 to double
1256   %ftmp929 = sitofp i64 %b926 to double
1257   %tmp930 = fcmp olt double %ftmp929, %ftmp928
1258   %booltmp931 = icmp ne i1 %tmp930, false
1259   br i1 %booltmp931, label %while_body841, label %merge932
1260
1261 while_body841:                                    ; preds = %while840
1262   %a842 = load i64, i64* %a791
1263   %free_alloc843 = alloca i64
1264   store i64 %a842, i64* %free_alloc843
1265   %_tgt_header_844 = load i64*, i64** %tgt_header788
1266   %glr845 = load i64, i64* %free_alloc843
1267   %offsetload846 = load i64, i64* %_tgt_header_844
```

```
  %plus1847 = add i64 %offsetload846, 1
  %intcast848 = ptrtoint i64* %_tgt_header_844 to i64
  %offsetcalc849 = mul i64 %plus1847, 8
  %pmath850 = add i64 %intcast848, %offsetcalc849
  %pointercast851 = inttoptr i64 %pmath850 to i64*
  %rootload852 = load i64, i64* %pointercast851
  %offsetcalc853 = mul i64 %glr845, 8
  %pmath854 = add i64 %rootload852, %offsetcalc853
  %pointercast855 = inttoptr i64 %pmath854 to i64*
  %deref856 = load i64, i64* %pointercast855
  %derefpntr857 = inttoptr i64 %deref856 to i64*
  %fpaccess858 = bitcast i64* %derefpntr857 to double*
  %fpret859 = load double, double* %fpaccess858
  %a860 = load i64, i64* %a791
  %free_alloc861 = alloca i64
  store i64 %a860, i64* %free_alloc861
  %b862 = load i64, i64* %b792
  %free_alloc863 = alloca i64
  store i64 %b862, i64* %free_alloc863
  %_s1_header_864 = load i64*, i64** %s1_header789
  %glr865 = load i64, i64* %free_alloc863
  %glr866 = load i64, i64* %free_alloc861
  %offsetload867 = load i64, i64* %_s1_header_864
  %plus1868 = add i64 %offsetload867, 1
  %intcast869 = ptrtoint i64* %_s1_header_864 to i64
  %offsetcalc870 = mul i64 %plus1868, 8
  %pmath871 = add i64 %intcast869, %offsetcalc870
  %pointercast872 = inttoptr i64 %pmath871 to i64*
  %rootload873 = load i64, i64* %pointercast872
  %offsetcalc874 = mul i64 %glr866, 8
  %pmath875 = add i64 %rootload873, %offsetcalc874
  %pointercast876 = inttoptr i64 %pmath875 to i64*
  %deref877 = load i64, i64* %pointercast876
  %derefpntr878 = inttoptr i64 %deref877 to i64*
  %intcast879 = ptrtoint i64* %derefpntr878 to i64
  %offsetcalc880 = mul i64 %glr865, 8
  %pmath881 = add i64 %intcast879, %offsetcalc880
  %pointercast882 = inttoptr i64 %pmath881 to i64*
  %deref883 = load i64, i64* %pointercast882
  %derefpntr884 = inttoptr i64 %deref883 to i64*
  %fpaccess885 = bitcast i64* %derefpntr884 to double*
  %fpret886 = load double, double* %fpaccess885
  %b887 = load i64, i64* %b792
  %free_alloc888 = alloca i64
  store i64 %b887, i64* %free_alloc888
  %_s2_header_889 = load i64*, i64** %s2_header790
  %glr890 = load i64, i64* %free_alloc888
  %offsetload891 = load i64, i64* %_s2_header_889
  %plus1892 = add i64 %offsetload891, 1
  %intcast893 = ptrtoint i64* %_s2_header_889 to i64
  %offsetcalc894 = mul i64 %plus1892, 8
  %pmath895 = add i64 %intcast893, %offsetcalc894
  %pointercast896 = inttoptr i64 %pmath895 to i64*
  %rootload897 = load i64, i64* %pointercast896
  %offsetcalc898 = mul i64 %glr890, 8
  %pmath899 = add i64 %rootload897, %offsetcalc898
```

```
  %pointercast900 = inttoptr i64 %pmath899 to i64*
  %deref901 = load i64, i64* %pointercast900
  %derefpntr902 = inttoptr i64 %deref901 to i64*
  %fpaccess903 = bitcast i64* %derefpntr902 to double*
  %fpret904 = load double, double* %fpaccess903
  %tmp905 = fmul double %fpret886, %fpret904
  %tmp906 = fadd double %fpret859, %tmp905
  %a907 = load i64, i64* %a791
  %free_alloc908 = alloca i64
  store i64 %a907, i64* %free_alloc908
  %_tgt_header_909 = load i64*, i64** %tgt_header788
  %glr910 = load i64, i64* %free_alloc908
  %offsetload911 = load i64, i64* %_tgt_header_909
  %plus1912 = add i64 %offsetload911, 1
  %intcast913 = ptrtoint i64* %_tgt_header_909 to i64
  %offsetcalc914 = mul i64 %plus1912, 8
  %pmath915 = add i64 %intcast913, %offsetcalc914
  %pointercast916 = inttoptr i64 %pmath915 to i64*
  %rootload917 = load i64, i64* %pointercast916
  %offsetcalc918 = mul i64 %glr910, 8
  %pmath919 = add i64 %rootload917, %offsetcalc918
  %pointercast920 = inttoptr i64 %pmath919 to i64*
  %deref921 = load i64, i64* %pointercast920
  %derefpntr922 = inttoptr i64 %deref921 to i64*
  %fpaccess923 = bitcast i64* %derefpntr922 to double*
  store double %tmp906, double* %fpaccess923
  %b924 = load i64, i64* %b792
  %tmp925 = add i64 1, %b924
  store i64 %tmp925, i64* %b792
  br label %while840

merge932:                                         ; preds = %while840
  %a933 = load i64, i64* %a791
  %tmp934 = add i64 1, %a933
  store i64 %tmp934, i64* %a791
  br label %while807

merge941:                                         ; preds = %while807
  %hdraddr_load942 = load i64*, i64** %tgt_header788
  store i64* %hdraddr_load942, i64** %D
  %i943 = load i64, i64* %i763
  %tmp944 = add i64 %i943, 1
  store i64 %tmp944, i64* %i763
  br label %while764

merge949:                                         ; preds = %while764
  %printf_ignore950 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([17 x i8], [17 x
      i8]* @"after 42 years: ", i32 0, i32 0))
  %D951 = load i64*, i64** %D
  %_printf_prefix_1952 = alloca i64
  %intcast953 = ptrtoint i64* %D951 to i64
  %pmath954 = add i64 %intcast953, 8
  %pointercast955 = inttoptr i64 %pmath954 to i64*
  store i64 0, i64* %_printf_prefix_1952
  br label %while956

```

```
while956:                                         ; preds = %while_body957, %merge949
  %_printf_prefix_1976 = load i64, i64* %_printf_prefix_1952
  %_slicesize_0977 = load i64, i64* %pointercast955
  %ftmp978 = sitofp i64 %_slicesize_0977 to double
  %ftmp979 = sitofp i64 %_printf_prefix_1976 to double
  %tmp980 = fcmp olt double %ftmp979, %ftmp978
  %booltmp981 = icmp ne i1 %tmp980, false
  br i1 %booltmp981, label %while_body957, label %merge982

while_body957:                                    ; preds = %while956
  %gl958 = load i64, i64* %_printf_prefix_1952
  %offsetload959 = load i64, i64* %D951
  %plus1960 = add i64 %offsetload959, 1
  %intcast961 = ptrtoint i64* %D951 to i64
  %offsetcalc962 = mul i64 %plus1960, 8
  %pmath963 = add i64 %intcast961, %offsetcalc962
  %pointercast964 = inttoptr i64 %pmath963 to i64*
  %rootload965 = load i64, i64* %pointercast964
  %offsetcalc966 = mul i64 %gl958, 8
  %pmath967 = add i64 %rootload965, %offsetcalc966
  %pointercast968 = inttoptr i64 %pmath967 to i64*
  %deref969 = load i64, i64* %pointercast968
  %derefpntr970 = inttoptr i64 %deref969 to i64*
  %fpaccess971 = bitcast i64* %derefpntr970 to double*
  %fpret972 = load double, double* %fpaccess971
  %printf_ignore973 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([7 x i8], [7 x i8]*
      @fmt3, i32 0, i32 0), double %fpret972)
  %_printf_prefix_1974 = load i64, i64* %_printf_prefix_1952
  %tmp975 = add i64 1, %_printf_prefix_1974
  store i64 %tmp975, i64* %_printf_prefix_1952
  br label %while956

merge982:                                         ; preds = %while956
  %printf_ignore983 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([2 x i8], [2 x i8]*
      @nlstr, i32 0, i32 0))
  ret i64 0
}
```

We can also pretty-print the code into LaTeX using the command `./latens.native -p demos/demo1.tens`.

```
A pithy code sample that shows us
how much cattle we would have if we
bought 40 cattle some number of years ago.
With the help of our friends: https://www.wku.edu/mathmatters/documents/mathmattersep13.pdf
(ie we implemented their stuff in LaTenS)
int main()
{
    print("Suppose you purchase 40 head of 2 year old cattle ");
    Our Leslie Matrix, for heads of cattle
```

```
    let L = ⎛ 0    0.5  1.1  1.3  0.8  0.4 ⎞
            ⎜ 0.7  0    0    0    0    0   ⎟
            ⎜ 0    0.9  0    0    0    0   ⎟
            ⎜ 0    0    0.8  0    0    0   ⎟
            ⎜ 0    0    0    0.7  0    0   ⎟
            ⎝ 0    0    0    0    0.3  0   ⎠ ;
    We've got 40 cattle at t=0; where t is measured in years
    let D0 = ⎛0  40  0  0  0  0⎞;

    print("given survival rates and birth rates: ");

    print(L);

    print("you want to know what your age distribution is ");

    print("after 2 years: ");
    Each multiplication moves t forward 2 years.
    So now t=2
    let D = L_{a,b} · D0_b;

    print(D);
    Let's jump t forward 28 more years
    So t = 2 + 28 = 30
    for (let i = 0 ; i < 14 ; i = (i + 1))
    {
        D = L_{a,b} · D_b;

    }

    print("after 30 years: ");

    print(D);
    And of course, no demo would be complete
    without showing t=42
    for (let i = 0 ; i < 26 ; i = (i + 1))
    {
        D = L_{a,b} · D_b;

    }

    print("after 42 years: ");

    print(D);

    return 0;
}
```

7.1.2 Example: Time Dilation on GPS Satellite Demo

The second demo is a general relativity application that calulates the difference in time measured by GPS Satellites to time measured on the surface of the earth due to general and special relativity. This application includes the standard library.
The LaTenS source code:

```
    let M = 5.972e24;
    let G = 6.67408e-11;
    let c = 2.99e8;
    let v = 3874.;
    let re = 6.357e6;
    let o = 2.6541e7;
    let sperday = 86400.;


int main () {
   let r = re+o;

   let S_{4, 4};
   for(let i = 0; i<4; i = i+1){
       for(let j = 0; j<4; j=j+1){
           S_{i, j} = 0;
        }
    }

    let x=2*G*M/(r*pow(c, 2));

    S_{1, 1} = 1/sqrt(1-x); %1+x/2+3/8*pow(x, 2)+5/16*pow(x, 3);
    S_{0, 0} = -1/S_{1,1};
    S_{2, 2} = S_{3, 3} = -r;

   print("\nOver the course of one day, ");


   getDilation(sperday, "day");
   print("s/day\n");


   print("over the course of one week, ");

   getDilation(sperday*7, "week");


   print("over the course of 42 days, ");

   getDilation(sperday*42, "42 days");


    return 0;
}

void getDilation(float t, string name){
    let g = v*v/(c*c);
    let gg = G*M/(re*pow(c, 2))-G*M/(o*pow(c, 2));

```

```
    let ts = t*(1.+g/2.+3./8.*pow(g, 2)+5./16.*pow(g, 3)) -t;
    let tg = t*gg;

   print("you lose " ^ ftos(ts) ^ " s from special relativity and gain " ^ ftos(tg) ^ " from
       general relativity\n");

   let err = -ts+tg;
   print(err);
   print("so a gps clock would find " ^ ftos(t+err) ^ "s/" ^ name);
   print(" which results in an error of " ^ ftos(err*c) ^ "m/" ^ name ^ "\n");
}
```

And the LLVM IR:

```
; ModuleID = 'LaTenS'

@G = global double 6.674080e-11
@M = global double 5.972000e+24
@c = global double 2.990000e+08
@o = global double 2.654100e+07
@re = global double 6.357000e+06
@sperday = global double 8.640000e+04
@v = global double 3.874000e+03
@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt = private unnamed_addr constant [5 x i8] c"%.6f\00"
@"\0AOver the course of one day, " = private unnamed_addr constant [30 x i8] c"\0AOver the course
    of one day, \00"
@day = private unnamed_addr constant [4 x i8] c"day\00"
@"s/day\0A" = private unnamed_addr constant [7 x i8] c"s/day\0A\00"
@"over the course of one week, " = private unnamed_addr constant [30 x i8] c"over the course of
    one week, \00"
@week = private unnamed_addr constant [5 x i8] c"week\00"
@"over the course of 42 days, " = private unnamed_addr constant [29 x i8] c"over the course of 42
    days, \00"
@"42 days" = private unnamed_addr constant [8 x i8] c"42 days\00"
@fmt.1 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.2 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.3 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.4 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.5 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.6 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.7 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@"you lose " = private unnamed_addr constant [10 x i8] c"you lose \00"
@" s from special relativity and gain " = private unnamed_addr constant [37 x i8] c" s from
    special relativity and gain \00"
@" from general relativity\0A" = private unnamed_addr constant [26 x i8] c" from general
    relativity\0A\00"
@"so a gps clock would find " = private unnamed_addr constant [27 x i8] c"so a gps clock would
    find \00"
@"s/" = private unnamed_addr constant [3 x i8] c"s/\00"
@" which results in an error of " = private unnamed_addr constant [31 x i8] c" which results in an
```

```
    error of \00"
@"m/" = private unnamed_addr constant [3 x i8] c"m/\00"
@"\0A" = private unnamed_addr constant [2 x i8] c"\0A\00"
@fmt.8 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.9 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.10 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.11 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.12 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.13 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.14 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.15 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.16 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.17 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.18 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.19 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.20 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.21 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.22 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.23 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.24 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.25 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.26 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.27 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.28 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.29 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.30 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.31 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.32 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.33 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.34 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.35 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.36 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.37 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.38 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.39 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.40 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.41 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.42 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.43 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.44 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.45 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.46 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.47 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.48 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.49 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.50 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.51 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.52 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.53 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.54 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.55 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.56 = private unnamed_addr constant [5 x i8] c"%.6f\00"

declare i64 @printf(i8*, ...)

declare i32 @strlen(i8*)
```

```
92
93  declare i64 @sprintf(i8*, i8*, ...)
94
95  declare i64 @atoi(i8*)
96
97  declare double @atof(i8*)
98
99  declare i64 @fopen(i8*, i8*)
100
101 declare i64 @fclose(i64)
102
103 declare i32 @fread(i64*, i32, i32, i64*)
104
105 declare i64 @fwrite(i64*, i32, i32, i64*)
106
107 declare double @sqrt(double)
108
109 define i64 @main() {
110 entry:
111   %r = alloca double
112   %re = load double, double* @re
113   %o = load double, double* @o
114   %tmp = fadd double %re, %o
115   store double %tmp, double* %r
116   %0 = trunc i64 4 to i32
117   %mallocsize = mul i32 %0, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
118   %malloccall = tail call i8* @malloc(i32 %mallocsize)
119   %dynamic_ten = bitcast i8* %malloccall to i64*
120   store i64 2, i64* %dynamic_ten
121   %intcast = ptrtoint i64* %dynamic_ten to i64
122   %pmath = add i64 %intcast, 8
123   %pointercast = inttoptr i64 %pmath to i64*
124   store i64 4, i64* %pointercast
125   %intcast1 = ptrtoint i64* %dynamic_ten to i64
126   %pmath2 = add i64 %intcast1, 16
127   %pointercast3 = inttoptr i64 %pmath2 to i64*
128   store i64 4, i64* %pointercast3
129   %_free_vars2 = alloca i64
130   %_free_vars1 = alloca i64
131   %intcast4 = ptrtoint i64* %dynamic_ten to i64
132   %pmath5 = add i64 %intcast4, 8
133   %pointercast6 = inttoptr i64 %pmath5 to i64*
134   %slicesizeload = load i64, i64* %pointercast6
135   %offsetload = load i64, i64* %dynamic_ten
136   %plus1 = add i64 %offsetload, 1
137   %intcast7 = ptrtoint i64* %dynamic_ten to i64
138   %offsetcalc = mul i64 %plus1, 8
139   %pmath8 = add i64 %intcast7, %offsetcalc
140   %pointercast9 = inttoptr i64 %pmath8 to i64*
141   %1 = trunc i64 %slicesizeload to i32
142   %mallocsize10 = mul i32 %1, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
143   %malloccall11 = tail call i8* @malloc(i32 %mallocsize10)
144   %mcarr = bitcast i8* %malloccall11 to i64*
145   %mcarrptrtoint = ptrtoint i64* %mcarr to i64
146   store i64 %mcarrptrtoint, i64* %pointercast9
147   store i64 0, i64* %_free_vars2
```

```
  br label %while

while:                                            ; preds = %merge, %entry
  %_free_vars258 = load i64, i64* %_free_vars2
  %_slicesize_0 = load i64, i64* %pointercast6
  %ftmp59 = sitofp i64 %_slicesize_0 to double
  %ftmp60 = sitofp i64 %_free_vars258 to double
  %tmp61 = fcmp olt double %ftmp60, %ftmp59
  %booltmp62 = icmp ne i1 %tmp61, false
  br i1 %booltmp62, label %while_body, label %merge63

while_body:                                       ; preds = %while
  %gl = load i64, i64* %_free_vars2
  %intcast12 = ptrtoint i64* %dynamic_ten to i64
  %pmath13 = add i64 %intcast12, 8
  %pointercast14 = inttoptr i64 %pmath13 to i64*
  %intcast15 = ptrtoint i64* %dynamic_ten to i64
  %pmath16 = add i64 %intcast15, 16
  %pointercast17 = inttoptr i64 %pmath16 to i64*
  %slicesizeload18 = load i64, i64* %pointercast17
  %offsetload19 = load i64, i64* %dynamic_ten
  %plus120 = add i64 %offsetload19, 1
  %intcast21 = ptrtoint i64* %dynamic_ten to i64
  %offsetcalc22 = mul i64 %plus120, 8
  %pmath23 = add i64 %intcast21, %offsetcalc22
  %pointercast24 = inttoptr i64 %pmath23 to i64*
  %deref = load i64, i64* %pointercast24
  %offsetcalc25 = mul i64 %gl, 8
  %pmath26 = add i64 %deref, %offsetcalc25
  %pointercast27 = inttoptr i64 %pmath26 to i64*
  %2 = trunc i64 %slicesizeload18 to i32
  %mallocsize28 = mul i32 %2, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall29 = tail call i8* @malloc(i32 %mallocsize28)
  %mcarr30 = bitcast i8* %malloccall29 to i64*
  %mcarrptrtoint31 = ptrtoint i64* %mcarr30 to i64
  store i64 %mcarrptrtoint31, i64* %pointercast27
  store i64 0, i64* %_free_vars1
  br label %while32

while32:                                          ; preds = %while_body33, %while_body
  %_free_vars153 = load i64, i64* %_free_vars1
  %_slicesize_1 = load i64, i64* %pointercast17
  %ftmp = sitofp i64 %_slicesize_1 to double
  %ftmp54 = sitofp i64 %_free_vars153 to double
  %tmp55 = fcmp olt double %ftmp54, %ftmp
  %booltmp = icmp ne i1 %tmp55, false
  br i1 %booltmp, label %while_body33, label %merge

while_body33:                                     ; preds = %while32
  %gl34 = load i64, i64* %_free_vars1
  %gl35 = load i64, i64* %_free_vars2
  %offsetload36 = load i64, i64* %dynamic_ten
  %plus137 = add i64 %offsetload36, 1
  %intcast38 = ptrtoint i64* %dynamic_ten to i64
  %offsetcalc39 = mul i64 %plus137, 8
  %pmath40 = add i64 %intcast38, %offsetcalc39
```

```
  %pointercast41 = inttoptr i64 %pmath40 to i64*
  %deref42 = load i64, i64* %pointercast41
  %offsetcalc43 = mul i64 %gl35, 8
  %pmath44 = add i64 %deref42, %offsetcalc43
  %pointercast45 = inttoptr i64 %pmath44 to i64*
  %deref46 = load i64, i64* %pointercast45
  %offsetcalc47 = mul i64 %gl34, 8
  %pmath48 = add i64 %deref46, %offsetcalc47
  %pointercast49 = inttoptr i64 %pmath48 to i64*
  %malloccall50 = tail call i8* @malloc(i32 ptrtoint (i64* getelementptr (i64, i64* null, i32 1)
      to i32))
  %abldrk = bitcast i8* %malloccall50 to i64*
  %abldrkcast = ptrtoint i64* %abldrk to i64
  store i64 %abldrkcast, i64* %pointercast49
  %_free_vars151 = load i64, i64* %_free_vars1
  %tmp52 = add i64 1, %_free_vars151
  store i64 %tmp52, i64* %_free_vars1
  br label %while32

merge:                                            ; preds = %while32
  %_free_vars256 = load i64, i64* %_free_vars2
  %tmp57 = add i64 1, %_free_vars256
  store i64 %tmp57, i64* %_free_vars2
  br label %while

merge63:                                          ; preds = %while
  %S = alloca i64*
  store i64* %dynamic_ten, i64** %S
  %i = alloca i64
  store i64 0, i64* %i
  br label %while64

while64:                                          ; preds = %merge95, %merge63
  %i98 = load i64, i64* %i
  %ftmp99 = sitofp i64 %i98 to double
  %tmp100 = fcmp olt double %ftmp99, 4.000000e+00
  %booltmp101 = icmp ne i1 %tmp100, false
  br i1 %booltmp101, label %while_body65, label %merge102

while_body65:                                     ; preds = %while64
  %j = alloca i64
  store i64 0, i64* %j
  br label %while66

while66:                                          ; preds = %while_body67, %while_body65
  %j91 = load i64, i64* %j
  %ftmp92 = sitofp i64 %j91 to double
  %tmp93 = fcmp olt double %ftmp92, 4.000000e+00
  %booltmp94 = icmp ne i1 %tmp93, false
  br i1 %booltmp94, label %while_body67, label %merge95

while_body67:                                     ; preds = %while66
  %i68 = load i64, i64* %i
  %free_alloc = alloca i64
  store i64 %i68, i64* %free_alloc
  %j69 = load i64, i64* %j
```

```
  %free_alloc70 = alloca i64
  store i64 %j69, i64* %free_alloc70
  %S71 = load i64*, i64** %S
  %glr = load i64, i64* %free_alloc70
  %glr72 = load i64, i64* %free_alloc
  %offsetload73 = load i64, i64* %S71
  %plus174 = add i64 %offsetload73, 1
  %intcast75 = ptrtoint i64* %S71 to i64
  %offsetcalc76 = mul i64 %plus174, 8
  %pmath77 = add i64 %intcast75, %offsetcalc76
  %pointercast78 = inttoptr i64 %pmath77 to i64*
  %rootload = load i64, i64* %pointercast78
  %offsetcalc79 = mul i64 %glr72, 8
  %pmath80 = add i64 %rootload, %offsetcalc79
  %pointercast81 = inttoptr i64 %pmath80 to i64*
  %deref82 = load i64, i64* %pointercast81
  %derefpntr = inttoptr i64 %deref82 to i64*
  %intcast83 = ptrtoint i64* %derefpntr to i64
  %offsetcalc84 = mul i64 %glr, 8
  %pmath85 = add i64 %intcast83, %offsetcalc84
  %pointercast86 = inttoptr i64 %pmath85 to i64*
  %deref87 = load i64, i64* %pointercast86
  %derefpntr88 = inttoptr i64 %deref87 to i64*
  %fpaccess = bitcast i64* %derefpntr88 to double*
  store double 0.000000e+00, double* %fpaccess
  %j89 = load i64, i64* %j
  %tmp90 = add i64 %j89, 1
  store i64 %tmp90, i64* %j
  br label %while66

merge95:                                          ; preds = %while66
  %i96 = load i64, i64* %i
  %tmp97 = add i64 %i96, 1
  store i64 %tmp97, i64* %i
  br label %while64

merge102:                                         ; preds = %while64
  %x = alloca double
  %G = load double, double* @G
  %tmp103 = fmul double 2.000000e+00, %G
  %M = load double, double* @M
  %tmp104 = fmul double %tmp103, %M
  %r105 = load double, double* %r
  %c = load double, double* @c
  %pow_result = call double @pow(double %c, i64 2)
  %tmp106 = fmul double %r105, %pow_result
  %tmp107 = fdiv double %tmp104, %tmp106
  store double %tmp107, double* %x
  %x108 = load double, double* %x
  %tmp109 = fsub double 1.000000e+00, %x108
  %sqrt_tmp = call double @sqrt(double %tmp109)
  %tmp110 = fdiv double 1.000000e+00, %sqrt_tmp
  %bnd_alloc = alloca i64
  store i64 1, i64* %bnd_alloc
  %bnd_alloc111 = alloca i64
  store i64 1, i64* %bnd_alloc111
```

```
  %S112 = load i64*, i64** %S
  %glr113 = load i64, i64* %bnd_alloc111
  %glr114 = load i64, i64* %bnd_alloc
  %offsetload115 = load i64, i64* %S112
  %plus1116 = add i64 %offsetload115, 1
  %intcast117 = ptrtoint i64* %S112 to i64
  %offsetcalc118 = mul i64 %plus1116, 8
  %pmath119 = add i64 %intcast117, %offsetcalc118
  %pointercast120 = inttoptr i64 %pmath119 to i64*
  %rootload121 = load i64, i64* %pointercast120
  %offsetcalc122 = mul i64 %glr114, 8
  %pmath123 = add i64 %rootload121, %offsetcalc122
  %pointercast124 = inttoptr i64 %pmath123 to i64*
  %deref125 = load i64, i64* %pointercast124
  %derefpntr126 = inttoptr i64 %deref125 to i64*
  %intcast127 = ptrtoint i64* %derefpntr126 to i64
  %offsetcalc128 = mul i64 %glr113, 8
  %pmath129 = add i64 %intcast127, %offsetcalc128
  %pointercast130 = inttoptr i64 %pmath129 to i64*
  %deref131 = load i64, i64* %pointercast130
  %derefpntr132 = inttoptr i64 %deref131 to i64*
  %fpaccess133 = bitcast i64* %derefpntr132 to double*
  store double %tmp110, double* %fpaccess133
  %bnd_alloc134 = alloca i64
  store i64 1, i64* %bnd_alloc134
  %bnd_alloc135 = alloca i64
  store i64 1, i64* %bnd_alloc135
  %S136 = load i64*, i64** %S
  %glr137 = load i64, i64* %bnd_alloc135
  %glr138 = load i64, i64* %bnd_alloc134
  %offsetload139 = load i64, i64* %S136
  %plus1140 = add i64 %offsetload139, 1
  %intcast141 = ptrtoint i64* %S136 to i64
  %offsetcalc142 = mul i64 %plus1140, 8
  %pmath143 = add i64 %intcast141, %offsetcalc142
  %pointercast144 = inttoptr i64 %pmath143 to i64*
  %rootload145 = load i64, i64* %pointercast144
  %offsetcalc146 = mul i64 %glr138, 8
  %pmath147 = add i64 %rootload145, %offsetcalc146
  %pointercast148 = inttoptr i64 %pmath147 to i64*
  %deref149 = load i64, i64* %pointercast148
  %derefpntr150 = inttoptr i64 %deref149 to i64*
  %intcast151 = ptrtoint i64* %derefpntr150 to i64
  %offsetcalc152 = mul i64 %glr137, 8
  %pmath153 = add i64 %intcast151, %offsetcalc152
  %pointercast154 = inttoptr i64 %pmath153 to i64*
  %deref155 = load i64, i64* %pointercast154
  %derefpntr156 = inttoptr i64 %deref155 to i64*
  %fpaccess157 = bitcast i64* %derefpntr156 to double*
  %fpret = load double, double* %fpaccess157
  %tmp158 = fdiv double -1.000000e+00, %fpret
  %bnd_alloc159 = alloca i64
  store i64 0, i64* %bnd_alloc159
  %bnd_alloc160 = alloca i64
  store i64 0, i64* %bnd_alloc160
  %S161 = load i64*, i64** %S
```

```
  %glr162 = load i64, i64* %bnd_alloc160
  %glr163 = load i64, i64* %bnd_alloc159
  %offsetload164 = load i64, i64* %S161
  %plus1165 = add i64 %offsetload164, 1
  %intcast166 = ptrtoint i64* %S161 to i64
  %offsetcalc167 = mul i64 %plus1165, 8
  %pmath168 = add i64 %intcast166, %offsetcalc167
  %pointercast169 = inttoptr i64 %pmath168 to i64*
  %rootload170 = load i64, i64* %pointercast169
  %offsetcalc171 = mul i64 %glr163, 8
  %pmath172 = add i64 %rootload170, %offsetcalc171
  %pointercast173 = inttoptr i64 %pmath172 to i64*
  %deref174 = load i64, i64* %pointercast173
  %derefpntr175 = inttoptr i64 %deref174 to i64*
  %intcast176 = ptrtoint i64* %derefpntr175 to i64
  %offsetcalc177 = mul i64 %glr162, 8
  %pmath178 = add i64 %intcast176, %offsetcalc177
  %pointercast179 = inttoptr i64 %pmath178 to i64*
  %deref180 = load i64, i64* %pointercast179
  %derefpntr181 = inttoptr i64 %deref180 to i64*
  %fpaccess182 = bitcast i64* %derefpntr181 to double*
  store double %tmp158, double* %fpaccess182
  %r183 = load double, double* %r
  %tmp184 = fsub double 0.000000e+00, %r183
  %bnd_alloc185 = alloca i64
  store i64 3, i64* %bnd_alloc185
  %bnd_alloc186 = alloca i64
  store i64 3, i64* %bnd_alloc186
  %S187 = load i64*, i64** %S
  %glr188 = load i64, i64* %bnd_alloc186
  %glr189 = load i64, i64* %bnd_alloc185
  %offsetload190 = load i64, i64* %S187
  %plus1191 = add i64 %offsetload190, 1
  %intcast192 = ptrtoint i64* %S187 to i64
  %offsetcalc193 = mul i64 %plus1191, 8
  %pmath194 = add i64 %intcast192, %offsetcalc193
  %pointercast195 = inttoptr i64 %pmath194 to i64*
  %rootload196 = load i64, i64* %pointercast195
  %offsetcalc197 = mul i64 %glr189, 8
  %pmath198 = add i64 %rootload196, %offsetcalc197
  %pointercast199 = inttoptr i64 %pmath198 to i64*
  %deref200 = load i64, i64* %pointercast199
  %derefpntr201 = inttoptr i64 %deref200 to i64*
  %intcast202 = ptrtoint i64* %derefpntr201 to i64
  %offsetcalc203 = mul i64 %glr188, 8
  %pmath204 = add i64 %intcast202, %offsetcalc203
  %pointercast205 = inttoptr i64 %pmath204 to i64*
  %deref206 = load i64, i64* %pointercast205
  %derefpntr207 = inttoptr i64 %deref206 to i64*
  %fpaccess208 = bitcast i64* %derefpntr207 to double*
  store double %tmp184, double* %fpaccess208
  %bnd_alloc209 = alloca i64
  store i64 2, i64* %bnd_alloc209
  %bnd_alloc210 = alloca i64
  store i64 2, i64* %bnd_alloc210
  %S211 = load i64*, i64** %S
```

```
  %glr212 = load i64, i64* %bnd_alloc210
  %glr213 = load i64, i64* %bnd_alloc209
  %offsetload214 = load i64, i64* %S211
  %plus1215 = add i64 %offsetload214, 1
  %intcast216 = ptrtoint i64* %S211 to i64
  %offsetcalc217 = mul i64 %plus1215, 8
  %pmath218 = add i64 %intcast216, %offsetcalc217
  %pointercast219 = inttoptr i64 %pmath218 to i64*
  %rootload220 = load i64, i64* %pointercast219
  %offsetcalc221 = mul i64 %glr213, 8
  %pmath222 = add i64 %rootload220, %offsetcalc221
  %pointercast223 = inttoptr i64 %pmath222 to i64*
  %deref224 = load i64, i64* %pointercast223
  %derefpntr225 = inttoptr i64 %deref224 to i64*
  %intcast226 = ptrtoint i64* %derefpntr225 to i64
  %offsetcalc227 = mul i64 %glr212, 8
  %pmath228 = add i64 %intcast226, %offsetcalc227
  %pointercast229 = inttoptr i64 %pmath228 to i64*
  %deref230 = load i64, i64* %pointercast229
  %derefpntr231 = inttoptr i64 %deref230 to i64*
  %fpaccess232 = bitcast i64* %derefpntr231 to double*
  store double %tmp184, double* %fpaccess232
  %printf_ignore = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([30 x i8], [30 x i8]*
      @"\0AOver the course of one day, ", i32 0, i32 0))
  %sperday = load double, double* @sperday
  call void @getDilation(double %sperday, i8* getelementptr inbounds ([4 x i8], [4 x i8]* @day,
      i32 0, i32 0))
  %printf_ignore233 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([7 x i8], [7 x i8]*
      @"s/day\0A", i32 0, i32 0))
  %printf_ignore234 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([30 x i8], [30 x
      i8]* @"over the course of one week, ", i32 0, i32 0))
  %sperday235 = load double, double* @sperday
  %tmp236 = fmul double %sperday235, 7.000000e+00
  call void @getDilation(double %tmp236, i8* getelementptr inbounds ([5 x i8], [5 x i8]* @week,
      i32 0, i32 0))
  %printf_ignore237 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([29 x i8], [29 x
      i8]* @"over the course of 42 days, ", i32 0, i32 0))
  %sperday238 = load double, double* @sperday
  %tmp239 = fmul double %sperday238, 4.200000e+01
  call void @getDilation(double %tmp239, i8* getelementptr inbounds ([8 x i8], [8 x i8]* @"42
      days", i32 0, i32 0))
  ret i64 0
}

define void @getDilation(double %t, i8* %name) {
entry:
  %t1 = alloca double
  store double %t, double* %t1
  %name2 = alloca i8*
  store i8* %name, i8** %name2
  %g = alloca double
  %v = load double, double* @v
  %v3 = load double, double* @v
  %tmp = fmul double %v, %v3
  %c = load double, double* @c
  %c4 = load double, double* @c
```

```
476   %tmp5 = fmul double %c, %c4
477   %tmp6 = fdiv double %tmp, %tmp5
478   store double %tmp6, double* %g
479   %gg = alloca double
480   %G = load double, double* @G
481   %M = load double, double* @M
482   %tmp7 = fmul double %G, %M
483   %re = load double, double* @re
484   %c8 = load double, double* @c
485   %pow_result = call double @pow(double %c8, i64 2)
486   %tmp9 = fmul double %re, %pow_result
487   %tmp10 = fdiv double %tmp7, %tmp9
488   %G11 = load double, double* @G
489   %M12 = load double, double* @M
490   %tmp13 = fmul double %G11, %M12
491   %o = load double, double* @o
492   %c14 = load double, double* @c
493   %pow_result15 = call double @pow(double %c14, i64 2)
494   %tmp16 = fmul double %o, %pow_result15
495   %tmp17 = fdiv double %tmp13, %tmp16
496   %tmp18 = fsub double %tmp10, %tmp17
497   store double %tmp18, double* %gg
498   %ts = alloca double
499   %t19 = load double, double* %t1
500   %g20 = load double, double* %g
501   %tmp21 = fdiv double %g20, 2.000000e+00
502   %tmp22 = fadd double 1.000000e+00, %tmp21
503   %g23 = load double, double* %g
504   %pow_result24 = call double @pow(double %g23, i64 2)
505   %tmp25 = fmul double 3.750000e-01, %pow_result24
506   %tmp26 = fadd double %tmp22, %tmp25
507   %g27 = load double, double* %g
508   %pow_result28 = call double @pow(double %g27, i64 3)
509   %tmp29 = fmul double 3.125000e-01, %pow_result28
510   %tmp30 = fadd double %tmp26, %tmp29
511   %tmp31 = fmul double %t19, %tmp30
512   %t32 = load double, double* %t1
513   %tmp33 = fsub double %tmp31, %t32
514   store double %tmp33, double* %ts
515   %tg = alloca double
516   %t34 = load double, double* %t1
517   %gg35 = load double, double* %gg
518   %tmp36 = fmul double %t34, %gg35
519   store double %tmp36, double* %tg
520   %malloc_tmp = tail call i8* @malloc(i32 mul (i32 ptrtoint (i8* getelementptr (i8, i8* null, i32
          1) to i32), i32 50))
521   %ts37 = load double, double* %ts
522   %sprintf_tmp = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp, i8* getelementptr inbounds ([5
          x i8], [5 x i8]* @ftosfmt.7, i32 0, i32 0), double %ts37)
523   %strlen1 = call i32 @strlen(i8* getelementptr inbounds ([10 x i8], [10 x i8]* @"you lose ", i32
          0, i32 0))
524   %strlen2 = call i32 @strlen(i8* %malloc_tmp)
525   %add_tmp = add i32 %strlen1, %strlen2
526   %add_tmp38 = add i32 %add_tmp, 1
527   %mallocsize = mul i32 %add_tmp38, ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32)
528   %malloc_tmp39 = tail call i8* @malloc(i32 %mallocsize)
```

```
  %sprintf_tmp40 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp39, i8* getelementptr inbounds ([5 x i8], [5 x i8]* @concatfmt.4, i32 0, i32 0), i8* getelementptr inbounds ([10 x i8], [10 x i8]* @"you lose ", i32 0, i32 0), i8* %malloc_tmp)
  %strlen141 = call i32 @strlen(i8* %malloc_tmp39)
  %strlen242 = call i32 @strlen(i8* getelementptr inbounds ([37 x i8], [37 x i8]* @" s from special relativity and gain ", i32 0, i32 0))
  %add_tmp43 = add i32 %strlen141, %strlen242
  %add_tmp44 = add i32 %add_tmp43, 1
  %mallocsize45 = mul i32 %add_tmp44, ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32)
  %malloc_tmp46 = tail call i8* @malloc(i32 %mallocsize45)
  %sprintf_tmp47 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp46, i8* getelementptr inbounds ([5 x i8], [5 x i8]* @concatfmt.4, i32 0, i32 0), i8* %malloc_tmp39, i8* getelementptr inbounds ([37 x i8], [37 x i8]* @" s from special relativity and gain ", i32 0, i32 0))
  %malloc_tmp48 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32), i32 50))
  %tg49 = load double, double* %tg
  %sprintf_tmp50 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp48, i8* getelementptr inbounds ([5 x i8], [5 x i8]* @ftosfmt.7, i32 0, i32 0), double %tg49)
  %strlen151 = call i32 @strlen(i8* %malloc_tmp46)
  %strlen252 = call i32 @strlen(i8* %malloc_tmp48)
  %add_tmp53 = add i32 %strlen151, %strlen252
  %add_tmp54 = add i32 %add_tmp53, 1
  %mallocsize55 = mul i32 %add_tmp54, ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32)
  %malloc_tmp56 = tail call i8* @malloc(i32 %mallocsize55)
  %sprintf_tmp57 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp56, i8* getelementptr inbounds ([5 x i8], [5 x i8]* @concatfmt.4, i32 0, i32 0), i8* %malloc_tmp46, i8* %malloc_tmp48)
  %strlen158 = call i32 @strlen(i8* %malloc_tmp56)
  %strlen259 = call i32 @strlen(i8* getelementptr inbounds ([26 x i8], [26 x i8]* @" from general relativity\0A", i32 0, i32 0))
  %add_tmp60 = add i32 %strlen158, %strlen259
  %add_tmp61 = add i32 %add_tmp60, 1
  %mallocsize62 = mul i32 %add_tmp61, ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32)
  %malloc_tmp63 = tail call i8* @malloc(i32 %mallocsize62)
  %sprintf_tmp64 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp63, i8* getelementptr inbounds ([5 x i8], [5 x i8]* @concatfmt.4, i32 0, i32 0), i8* %malloc_tmp56, i8* getelementptr inbounds ([26 x i8], [26 x i8]* @" from general relativity\0A", i32 0, i32 0))
  %printf_ignore = call i64 (i8*, ...) @printf(i8* %malloc_tmp63)
  %err = alloca double
  %ts65 = load double, double* %ts
  %tmp66 = fsub double 0.000000e+00, %ts65
  %tg67 = load double, double* %tg
  %tmp68 = fadd double %tmp66, %tg67
  store double %tmp68, double* %err
  %err69 = load double, double* %err
  %printf_ignore70 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([6 x i8], [6 x i8]* @fmt2.2, i32 0, i32 0), double %err69)
  %malloc_tmp71 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32), i32 50))
  %t72 = load double, double* %t1
  %err73 = load double, double* %err
  %tmp74 = fadd double %t72, %err73
  %sprintf_tmp75 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp71, i8* getelementptr inbounds ([5 x i8], [5 x i8]* @ftosfmt.7, i32 0, i32 0), double %tmp74)
  %strlen176 = call i32 @strlen(i8* getelementptr inbounds ([27 x i8], [27 x i8]* @"so a gps clock would find ", i32 0, i32 0))
  %strlen277 = call i32 @strlen(i8* %malloc_tmp71)
```

```
  %add_tmp78 = add i32 %strlen176, %strlen277
  %add_tmp79 = add i32 %add_tmp78, 1
  %mallocsize80 = mul i32 %add_tmp79, ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32)
  %malloc_tmp81 = tail call i8* @malloc(i32 %mallocsize80)
  %sprintf_tmp82 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp81, i8* getelementptr inbounds ([5 x i8], [5 x i8]* @concatfmt.4, i32 0, i32 0), i8* getelementptr inbounds ([27 x i8], [27 x i8]* @"so a gps clock would find ", i32 0, i32 0), i8* %malloc_tmp71)
  %strlen183 = call i32 @strlen(i8* %malloc_tmp81)
  %strlen284 = call i32 @strlen(i8* getelementptr inbounds ([3 x i8], [3 x i8]* @"s/", i32 0, i32 0))
  %add_tmp85 = add i32 %strlen183, %strlen284
  %add_tmp86 = add i32 %add_tmp85, 1
  %mallocsize87 = mul i32 %add_tmp86, ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32)
  %malloc_tmp88 = tail call i8* @malloc(i32 %mallocsize87)
  %sprintf_tmp89 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp88, i8* getelementptr inbounds ([5 x i8], [5 x i8]* @concatfmt.4, i32 0, i32 0), i8* %malloc_tmp81, i8* getelementptr inbounds ([3 x i8], [3 x i8]* @"s/", i32 0, i32 0))
  %name90 = load i8*, i8** %name2
  %strlen191 = call i32 @strlen(i8* %malloc_tmp88)
  %strlen292 = call i32 @strlen(i8* %name90)
  %add_tmp93 = add i32 %strlen191, %strlen292
  %add_tmp94 = add i32 %add_tmp93, 1
  %mallocsize95 = mul i32 %add_tmp94, ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32)
  %malloc_tmp96 = tail call i8* @malloc(i32 %mallocsize95)
  %sprintf_tmp97 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp96, i8* getelementptr inbounds ([5 x i8], [5 x i8]* @concatfmt.4, i32 0, i32 0), i8* %malloc_tmp88, i8* %name90)
  %printf_ignore98 = call i64 (i8*, ...) @printf(i8* %malloc_tmp96)
  %malloc_tmp99 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32), i32 50))
  %err100 = load double, double* %err
  %c101 = load double, double* @c
  %tmp102 = fmul double %err100, %c101
  %sprintf_tmp103 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp99, i8* getelementptr inbounds ([5 x i8], [5 x i8]* @ftosfmt.7, i32 0, i32 0), double %tmp102)
  %strlen1104 = call i32 @strlen(i8* getelementptr inbounds ([31 x i8], [31 x i8]* @" which results in an error of ", i32 0, i32 0))
  %strlen2105 = call i32 @strlen(i8* %malloc_tmp99)
  %add_tmp106 = add i32 %strlen1104, %strlen2105
  %add_tmp107 = add i32 %add_tmp106, 1
  %mallocsize108 = mul i32 %add_tmp107, ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32)
  %malloc_tmp109 = tail call i8* @malloc(i32 %mallocsize108)
  %sprintf_tmp110 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp109, i8* getelementptr inbounds ([5 x i8], [5 x i8]* @concatfmt.4, i32 0, i32 0), i8* getelementptr inbounds ([31 x i8], [31 x i8]* @" which results in an error of ", i32 0, i32 0), i8* %malloc_tmp99)
  %strlen1111 = call i32 @strlen(i8* %malloc_tmp109)
  %strlen2112 = call i32 @strlen(i8* getelementptr inbounds ([3 x i8], [3 x i8]* @"m/", i32 0, i32 0))
  %add_tmp113 = add i32 %strlen1111, %strlen2112
  %add_tmp114 = add i32 %add_tmp113, 1
  %mallocsize115 = mul i32 %add_tmp114, ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32)
  %malloc_tmp116 = tail call i8* @malloc(i32 %mallocsize115)
  %sprintf_tmp117 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp116, i8* getelementptr inbounds ([5 x i8], [5 x i8]* @concatfmt.4, i32 0, i32 0), i8* %malloc_tmp109, i8* getelementptr inbounds ([3 x i8], [3 x i8]* @"m/", i32 0, i32 0))
  %name118 = load i8*, i8** %name2
  %strlen1119 = call i32 @strlen(i8* %malloc_tmp116)
```

```
612   %strlen2120 = call i32 @strlen(i8* %name118)
613   %add_tmp121 = add i32 %strlen1119, %strlen2120
614   %add_tmp122 = add i32 %add_tmp121, 1
615   %mallocsize123 = mul i32 %add_tmp122, ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32)
616   %malloc_tmp124 = tail call i8* @malloc(i32 %mallocsize123)
617   %sprintf_tmp125 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp124, i8* getelementptr
          inbounds ([5 x i8], [5 x i8]* @concatfmt.4, i32 0, i32 0), i8* %malloc_tmp116, i8* %name118)
618   %strlen1126 = call i32 @strlen(i8* %malloc_tmp124)
619   %strlen2127 = call i32 @strlen(i8* getelementptr inbounds ([2 x i8], [2 x i8]* @"\0A", i32 0,
          i32 0))
620   %add_tmp128 = add i32 %strlen1126, %strlen2127
621   %add_tmp129 = add i32 %add_tmp128, 1
622   %mallocsize130 = mul i32 %add_tmp129, ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32)
623   %malloc_tmp131 = tail call i8* @malloc(i32 %mallocsize130)
624   %sprintf_tmp132 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp131, i8* getelementptr
          inbounds ([5 x i8], [5 x i8]* @concatfmt.4, i32 0, i32 0), i8* %malloc_tmp124, i8*
          getelementptr inbounds ([2 x i8], [2 x i8]* @"\0A", i32 0, i32 0))
625   %printf_ignore133 = call i64 (i8*, ...) @printf(i8* %malloc_tmp131)
626   ret void
627 }
628
629 define double @pow(double %f, i64 %i) {
630 entry:
631   %f1 = alloca double
632   store double %f, double* %f1
633   %i2 = alloca i64
634   store i64 %i, i64* %i2
635   %result = alloca double
636   %f3 = load double, double* %f1
637   store double %f3, double* %result
638   %j = alloca i64
639   store i64 1, i64* %j
640   br label %while
641
642 while:                                            ; preds = %while_body, %entry
643   %j8 = load i64, i64* %j
644   %i9 = load i64, i64* %i2
645   %ftmp = sitofp i64 %i9 to double
646   %ftmp10 = sitofp i64 %j8 to double
647   %tmp11 = fcmp olt double %ftmp10, %ftmp
648   %booltmp = icmp ne i1 %tmp11, false
649   br i1 %booltmp, label %while_body, label %merge
650
651 while_body:                                       ; preds = %while
652   %result4 = load double, double* %result
653   %f5 = load double, double* %f1
654   %tmp = fmul double %result4, %f5
655   store double %tmp, double* %result
656   %j6 = load i64, i64* %j
657   %tmp7 = add i64 %j6, 1
658   store i64 %tmp7, i64* %j
659   br label %while
660
661 merge:                                            ; preds = %while
662   %result12 = load double, double* %result
663   ret double %result12
```

```
}

define i64* @transpose(i64* %T) {
entry:
  %T1 = alloca i64*
  store i64* %T, i64** %T1
  %da = alloca i64
  %T2 = load i64*, i64** %T1
  %intcast = ptrtoint i64* %T2 to i64
  %pmath = add i64 %intcast, 8
  %pointercast = inttoptr i64 %pmath to i64*
  %tenresult = load i64, i64* %pointercast
  store i64 %tenresult, i64* %da
  %db = alloca i64
  %T3 = load i64*, i64** %T1
  %intcast4 = ptrtoint i64* %T3 to i64
  %pmath5 = add i64 %intcast4, 16
  %pointercast6 = inttoptr i64 %pmath5 to i64*
  %tenresult7 = load i64, i64* %pointercast6
  store i64 %tenresult7, i64* %db
  %0 = trunc i64 4 to i32
  %mallocsize = mul i32 %0, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall = tail call i8* @malloc(i32 %mallocsize)
  %dynamic_ten = bitcast i8* %malloccall to i64*
  store i64 2, i64* %dynamic_ten
  %da8 = load i64, i64* %da
  %db9 = load i64, i64* %db
  %intcast10 = ptrtoint i64* %dynamic_ten to i64
  %pmath11 = add i64 %intcast10, 8
  %pointercast12 = inttoptr i64 %pmath11 to i64*
  store i64 %da8, i64* %pointercast12
  %intcast13 = ptrtoint i64* %dynamic_ten to i64
  %pmath14 = add i64 %intcast13, 16
  %pointercast15 = inttoptr i64 %pmath14 to i64*
  store i64 %db9, i64* %pointercast15
  %_free_vars2 = alloca i64
  %_free_vars1 = alloca i64
  %intcast16 = ptrtoint i64* %dynamic_ten to i64
  %pmath17 = add i64 %intcast16, 8
  %pointercast18 = inttoptr i64 %pmath17 to i64*
  %slicesizeload = load i64, i64* %pointercast18
  %offsetload = load i64, i64* %dynamic_ten
  %plus1 = add i64 %offsetload, 1
  %intcast19 = ptrtoint i64* %dynamic_ten to i64
  %offsetcalc = mul i64 %plus1, 8
  %pmath20 = add i64 %intcast19, %offsetcalc
  %pointercast21 = inttoptr i64 %pmath20 to i64*
  %1 = trunc i64 %slicesizeload to i32
  %mallocsize22 = mul i32 %1, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall23 = tail call i8* @malloc(i32 %mallocsize22)
  %mcarr = bitcast i8* %malloccall23 to i64*
  %mcarrptrtoint = ptrtoint i64* %mcarr to i64
  store i64 %mcarrptrtoint, i64* %pointercast21
  store i64 0, i64* %_free_vars2
  br label %while

```

```
720 while:                                         ; preds = %merge, %entry
721   %_free_vars269 = load i64, i64* %_free_vars2
722   %_slicesize_0 = load i64, i64* %pointercast18
723   %ftmp70 = sitofp i64 %_slicesize_0 to double
724   %ftmp71 = sitofp i64 %_free_vars269 to double
725   %tmp72 = fcmp olt double %ftmp71, %ftmp70
726   %booltmp73 = icmp ne i1 %tmp72, false
727   br i1 %booltmp73, label %while_body, label %merge74
728
729 while_body:                                       ; preds = %while
730   %gl = load i64, i64* %_free_vars2
731   %intcast24 = ptrtoint i64* %dynamic_ten to i64
732   %pmath25 = add i64 %intcast24, 8
733   %pointercast26 = inttoptr i64 %pmath25 to i64*
734   %intcast27 = ptrtoint i64* %dynamic_ten to i64
735   %pmath28 = add i64 %intcast27, 16
736   %pointercast29 = inttoptr i64 %pmath28 to i64*
737   %slicesizeload30 = load i64, i64* %pointercast29
738   %offsetload31 = load i64, i64* %dynamic_ten
739   %plus132 = add i64 %offsetload31, 1
740   %intcast33 = ptrtoint i64* %dynamic_ten to i64
741   %offsetcalc34 = mul i64 %plus132, 8
742   %pmath35 = add i64 %intcast33, %offsetcalc34
743   %pointercast36 = inttoptr i64 %pmath35 to i64*
744   %deref = load i64, i64* %pointercast36
745   %offsetcalc37 = mul i64 %gl, 8
746   %pmath38 = add i64 %deref, %offsetcalc37
747   %pointercast39 = inttoptr i64 %pmath38 to i64*
748   %2 = trunc i64 %slicesizeload30 to i32
749   %mallocsize40 = mul i32 %2, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
750   %malloccall41 = tail call i8* @malloc(i32 %mallocsize40)
751   %mcarr42 = bitcast i8* %malloccall41 to i64*
752   %mcarrptrtoint43 = ptrtoint i64* %mcarr42 to i64
753   store i64 %mcarrptrtoint43, i64* %pointercast39
754   store i64 0, i64* %_free_vars1
755   br label %while44
756
757 while44:                                          ; preds = %while_body45, %while_body
758   %_free_vars164 = load i64, i64* %_free_vars1
759   %_slicesize_1 = load i64, i64* %pointercast29
760   %ftmp = sitofp i64 %_slicesize_1 to double
761   %ftmp65 = sitofp i64 %_free_vars164 to double
762   %tmp66 = fcmp olt double %ftmp65, %ftmp
763   %booltmp = icmp ne i1 %tmp66, false
764   br i1 %booltmp, label %while_body45, label %merge
765
766 while_body45:                                     ; preds = %while44
767   %gl146 = load i64, i64* %_free_vars1
768   %gl147 = load i64, i64* %_free_vars2
769   %offsetload48 = load i64, i64* %dynamic_ten
770   %plus149 = add i64 %offsetload48, 1
771   %intcast50 = ptrtoint i64* %dynamic_ten to i64
772   %offsetcalc51 = mul i64 %plus149, 8
773   %pmath52 = add i64 %intcast50, %offsetcalc51
774   %pointercast53 = inttoptr i64 %pmath52 to i64*
775   %deref54 = load i64, i64* %pointercast53
```

```
776   %offsetcalc55 = mul i64 %gl147, 8
777   %pmath56 = add i64 %deref54, %offsetcalc55
778   %pointercast57 = inttoptr i64 %pmath56 to i64*
779   %deref58 = load i64, i64* %pointercast57
780   %offsetcalc59 = mul i64 %gl146, 8
781   %pmath60 = add i64 %deref58, %offsetcalc59
782   %pointercast61 = inttoptr i64 %pmath60 to i64*
783   %malloccall62 = tail call i8* @malloc(i32 ptrtoint (i64* getelementptr (i64, i64* null, i32 1)
          to i32))
784   %abldrk = bitcast i8* %malloccall62 to i64*
785   %abldrkcast = ptrtoint i64* %abldrk to i64
786   store i64 %abldrkcast, i64* %pointercast61
787   %_free_vars163 = load i64, i64* %_free_vars1
788   %tmp = add i64 1, %_free_vars163
789   store i64 %tmp, i64* %_free_vars1
790   br label %while44
791
792 merge:                                            ; preds = %while44
793   %_free_vars267 = load i64, i64* %_free_vars2
794   %tmp68 = add i64 1, %_free_vars267
795   store i64 %tmp68, i64* %_free_vars2
796   br label %while
797
798 merge74:                                          ; preds = %while
799   %newT = alloca i64*
800   store i64* %dynamic_ten, i64** %newT
801   %i = alloca i64
802   store i64 0, i64* %i
803   br label %while75
804
805 while75:                                          ; preds = %merge134, %merge74
806   %i137 = load i64, i64* %i
807   %da138 = load i64, i64* %da
808   %ftmp139 = sitofp i64 %da138 to double
809   %ftmp140 = sitofp i64 %i137 to double
810   %tmp141 = fcmp olt double %ftmp140, %ftmp139
811   %booltmp142 = icmp ne i1 %tmp141, false
812   br i1 %booltmp142, label %while_body76, label %merge143
813
814 while_body76:                                     ; preds = %while75
815   %j = alloca i64
816   store i64 0, i64* %j
817   br label %while77
818
819 while77:                                          ; preds = %while_body78, %while_body76
820   %j128 = load i64, i64* %j
821   %db129 = load i64, i64* %db
822   %ftmp130 = sitofp i64 %db129 to double
823   %ftmp131 = sitofp i64 %j128 to double
824   %tmp132 = fcmp olt double %ftmp131, %ftmp130
825   %booltmp133 = icmp ne i1 %tmp132, false
826   br i1 %booltmp133, label %while_body78, label %merge134
827
828 while_body78:                                     ; preds = %while77
829   %j79 = load i64, i64* %j
830   %free_alloc = alloca i64
```

```
  store i64 %j79, i64* %free_alloc
  %i80 = load i64, i64* %i
  %free_alloc81 = alloca i64
  store i64 %i80, i64* %free_alloc81
  %T82 = load i64*, i64** %T1
  %glr = load i64, i64* %free_alloc81
  %glr83 = load i64, i64* %free_alloc
  %offsetload84 = load i64, i64* %T82
  %plus185 = add i64 %offsetload84, 1
  %intcast86 = ptrtoint i64* %T82 to i64
  %offsetcalc87 = mul i64 %plus185, 8
  %pmath88 = add i64 %intcast86, %offsetcalc87
  %pointercast89 = inttoptr i64 %pmath88 to i64*
  %rootload = load i64, i64* %pointercast89
  %offsetcalc90 = mul i64 %glr83, 8
  %pmath91 = add i64 %rootload, %offsetcalc90
  %pointercast92 = inttoptr i64 %pmath91 to i64*
  %deref93 = load i64, i64* %pointercast92
  %derefpntr = inttoptr i64 %deref93 to i64*
  %intcast94 = ptrtoint i64* %derefpntr to i64
  %offsetcalc95 = mul i64 %glr, 8
  %pmath96 = add i64 %intcast94, %offsetcalc95
  %pointercast97 = inttoptr i64 %pmath96 to i64*
  %deref98 = load i64, i64* %pointercast97
  %derefpntr99 = inttoptr i64 %deref98 to i64*
  %fpaccess = bitcast i64* %derefpntr99 to double*
  %fpret = load double, double* %fpaccess
  %i100 = load i64, i64* %i
  %free_alloc101 = alloca i64
  store i64 %i100, i64* %free_alloc101
  %j102 = load i64, i64* %j
  %free_alloc103 = alloca i64
  store i64 %j102, i64* %free_alloc103
  %newT104 = load i64*, i64** %newT
  %glr105 = load i64, i64* %free_alloc103
  %glr106 = load i64, i64* %free_alloc101
  %offsetload107 = load i64, i64* %newT104
  %plus1108 = add i64 %offsetload107, 1
  %intcast109 = ptrtoint i64* %newT104 to i64
  %offsetcalc110 = mul i64 %plus1108, 8
  %pmath111 = add i64 %intcast109, %offsetcalc110
  %pointercast112 = inttoptr i64 %pmath111 to i64*
  %rootload113 = load i64, i64* %pointercast112
  %offsetcalc114 = mul i64 %glr106, 8
  %pmath115 = add i64 %rootload113, %offsetcalc114
  %pointercast116 = inttoptr i64 %pmath115 to i64*
  %deref117 = load i64, i64* %pointercast116
  %derefpntr118 = inttoptr i64 %deref117 to i64*
  %intcast119 = ptrtoint i64* %derefpntr118 to i64
  %offsetcalc120 = mul i64 %glr105, 8
  %pmath121 = add i64 %intcast119, %offsetcalc120
  %pointercast122 = inttoptr i64 %pmath121 to i64*
  %deref123 = load i64, i64* %pointercast122
  %derefpntr124 = inttoptr i64 %deref123 to i64*
  %fpaccess125 = bitcast i64* %derefpntr124 to double*
  store double %fpret, double* %fpaccess125
```

```
  %j126 = load i64, i64* %j
  %tmp127 = add i64 %j126, 1
  store i64 %tmp127, i64* %j
  br label %while77

merge134:                                         ; preds = %while77
  %i135 = load i64, i64* %i
  %tmp136 = add i64 %i135, 1
  store i64 %tmp136, i64* %i
  br label %while75

merge143:                                         ; preds = %while75
  %newT144 = load i64*, i64** %newT
  ret i64* %newT144
}

define double @min(i64* %T) {
entry:
  %T1 = alloca i64*
  store i64* %T, i64** %T1
  %da = alloca i64
  %T2 = load i64*, i64** %T1
  %intcast = ptrtoint i64* %T2 to i64
  %pmath = add i64 %intcast, 8
  %pointercast = inttoptr i64 %pmath to i64*
  %tenresult = load i64, i64* %pointercast
  store i64 %tenresult, i64* %da
  %min = alloca double
  %bnd_alloc = alloca i64
  store i64 0, i64* %bnd_alloc
  %T3 = load i64*, i64** %T1
  %glr = load i64, i64* %bnd_alloc
  %offsetload = load i64, i64* %T3
  %plus1 = add i64 %offsetload, 1
  %intcast4 = ptrtoint i64* %T3 to i64
  %offsetcalc = mul i64 %plus1, 8
  %pmath5 = add i64 %intcast4, %offsetcalc
  %pointercast6 = inttoptr i64 %pmath5 to i64*
  %rootload = load i64, i64* %pointercast6
  %offsetcalc7 = mul i64 %glr, 8
  %pmath8 = add i64 %rootload, %offsetcalc7
  %pointercast9 = inttoptr i64 %pmath8 to i64*
  %deref = load i64, i64* %pointercast9
  %derefpntr = inttoptr i64 %deref to i64*
  %fpaccess = bitcast i64* %derefpntr to double*
  %fpret = load double, double* %fpaccess
  store double %fpret, double* %min
  %i = alloca i64
  store i64 1, i64* %i
  br label %while

while:                                            ; preds = %merge, %entry
  %i48 = load i64, i64* %i
  %da49 = load i64, i64* %da
  %ftmp = sitofp i64 %da49 to double
  %ftmp50 = sitofp i64 %i48 to double
```

```
  %tmp51 = fcmp olt double %ftmp50, %ftmp
  %booltmp52 = icmp ne i1 %tmp51, false
  br i1 %booltmp52, label %while_body, label %merge53

while_body:                                       ; preds = %while
  %i10 = load i64, i64* %i
  %free_alloc = alloca i64
  store i64 %i10, i64* %free_alloc
  %T11 = load i64*, i64** %T1
  %glr12 = load i64, i64* %free_alloc
  %offsetload13 = load i64, i64* %T11
  %plus114 = add i64 %offsetload13, 1
  %intcast15 = ptrtoint i64* %T11 to i64
  %offsetcalc16 = mul i64 %plus114, 8
  %pmath17 = add i64 %intcast15, %offsetcalc16
  %pointercast18 = inttoptr i64 %pmath17 to i64*
  %rootload19 = load i64, i64* %pointercast18
  %offsetcalc20 = mul i64 %glr12, 8
  %pmath21 = add i64 %rootload19, %offsetcalc20
  %pointercast22 = inttoptr i64 %pmath21 to i64*
  %deref23 = load i64, i64* %pointercast22
  %derefpntr24 = inttoptr i64 %deref23 to i64*
  %fpaccess25 = bitcast i64* %derefpntr24 to double*
  %fpret26 = load double, double* %fpaccess25
  %min27 = load double, double* %min
  %tmp = fcmp olt double %fpret26, %min27
  %booltmp = icmp ne i1 %tmp, false
  br i1 %booltmp, label %then, label %else

merge:                                            ; preds = %else, %then
  %i46 = load i64, i64* %i
  %tmp47 = add i64 %i46, 1
  store i64 %tmp47, i64* %i
  br label %while

then:                                             ; preds = %while_body
  %i28 = load i64, i64* %i
  %free_alloc29 = alloca i64
  store i64 %i28, i64* %free_alloc29
  %T30 = load i64*, i64** %T1
  %glr31 = load i64, i64* %free_alloc29
  %offsetload32 = load i64, i64* %T30
  %plus133 = add i64 %offsetload32, 1
  %intcast34 = ptrtoint i64* %T30 to i64
  %offsetcalc35 = mul i64 %plus133, 8
  %pmath36 = add i64 %intcast34, %offsetcalc35
  %pointercast37 = inttoptr i64 %pmath36 to i64*
  %rootload38 = load i64, i64* %pointercast37
  %offsetcalc39 = mul i64 %glr31, 8
  %pmath40 = add i64 %rootload38, %offsetcalc39
  %pointercast41 = inttoptr i64 %pmath40 to i64*
  %deref42 = load i64, i64* %pointercast41
  %derefpntr43 = inttoptr i64 %deref42 to i64*
  %fpaccess44 = bitcast i64* %derefpntr43 to double*
  %fpret45 = load double, double* %fpaccess44
  store double %fpret45, double* %min
```

```
  br label %merge

else:                                             ; preds = %while_body
  br label %merge

merge53:                                          ; preds = %while
  %min54 = load double, double* %min
  ret double %min54
}

define i64 @minInd(i64* %T) {
entry:
  %T1 = alloca i64*
  store i64* %T, i64** %T1
  %da = alloca i64
  %T2 = load i64*, i64** %T1
  %intcast = ptrtoint i64* %T2 to i64
  %pmath = add i64 %intcast, 8
  %pointercast = inttoptr i64 %pmath to i64*
  %tenresult = load i64, i64* %pointercast
  store i64 %tenresult, i64* %da
  %min = alloca i64
  store i64 0, i64* %min
  %i = alloca i64
  store i64 1, i64* %i
  br label %while

while:                                            ; preds = %merge, %entry
  %i32 = load i64, i64* %i
  %da33 = load i64, i64* %da
  %ftmp = sitofp i64 %da33 to double
  %ftmp34 = sitofp i64 %i32 to double
  %tmp35 = fcmp olt double %ftmp34, %ftmp
  %booltmp36 = icmp ne i1 %tmp35, false
  br i1 %booltmp36, label %while_body, label %merge37

while_body:                                       ; preds = %while
  %i3 = load i64, i64* %i
  %free_alloc = alloca i64
  store i64 %i3, i64* %free_alloc
  %T4 = load i64*, i64** %T1
  %glr = load i64, i64* %free_alloc
  %offsetload = load i64, i64* %T4
  %plus1 = add i64 %offsetload, 1
  %intcast5 = ptrtoint i64* %T4 to i64
  %offsetcalc = mul i64 %plus1, 8
  %pmath6 = add i64 %intcast5, %offsetcalc
  %pointercast7 = inttoptr i64 %pmath6 to i64*
  %rootload = load i64, i64* %pointercast7
  %offsetcalc8 = mul i64 %glr, 8
  %pmath9 = add i64 %rootload, %offsetcalc8
  %pointercast10 = inttoptr i64 %pmath9 to i64*
  %deref = load i64, i64* %pointercast10
  %derefpntr = inttoptr i64 %deref to i64*
  %fpaccess = bitcast i64* %derefpntr to double*
  %fpret = load double, double* %fpaccess
```

```
  %min11 = load i64, i64* %min
  %free_alloc12 = alloca i64
  store i64 %min11, i64* %free_alloc12
  %T13 = load i64*, i64** %T1
  %glr14 = load i64, i64* %free_alloc12
  %offsetload15 = load i64, i64* %T13
  %plus116 = add i64 %offsetload15, 1
  %intcast17 = ptrtoint i64* %T13 to i64
  %offsetcalc18 = mul i64 %plus116, 8
  %pmath19 = add i64 %intcast17, %offsetcalc18
  %pointercast20 = inttoptr i64 %pmath19 to i64*
  %rootload21 = load i64, i64* %pointercast20
  %offsetcalc22 = mul i64 %glr14, 8
  %pmath23 = add i64 %rootload21, %offsetcalc22
  %pointercast24 = inttoptr i64 %pmath23 to i64*
  %deref25 = load i64, i64* %pointercast24
  %derefpntr26 = inttoptr i64 %deref25 to i64*
  %fpaccess27 = bitcast i64* %derefpntr26 to double*
  %fpret28 = load double, double* %fpaccess27
  %tmp = fcmp olt double %fpret, %fpret28
  %booltmp = icmp ne i1 %tmp, false
  br i1 %booltmp, label %then, label %else

merge:                                            ; preds = %else, %then
  %i30 = load i64, i64* %i
  %tmp31 = add i64 %i30, 1
  store i64 %tmp31, i64* %i
  br label %while

then:                                             ; preds = %while_body
  %i29 = load i64, i64* %i
  store i64 %i29, i64* %min
  br label %merge

else:                                             ; preds = %while_body
  br label %merge

merge37:                                          ; preds = %while
  %min38 = load i64, i64* %min
  ret i64 %min38
}

define double @max(i64* %T) {
entry:
  %T1 = alloca i64*
  store i64* %T, i64** %T1
  %da = alloca i64
  %T2 = load i64*, i64** %T1
  %intcast = ptrtoint i64* %T2 to i64
  %pmath = add i64 %intcast, 8
  %pointercast = inttoptr i64 %pmath to i64*
  %tenresult = load i64, i64* %pointercast
  store i64 %tenresult, i64* %da
  %max = alloca double
  %bnd_alloc = alloca i64
  store i64 0, i64* %bnd_alloc
```

```
  %T3 = load i64*, i64** %T1
  %glr = load i64, i64* %bnd_alloc
  %offsetload = load i64, i64* %T3
  %plus1 = add i64 %offsetload, 1
  %intcast4 = ptrtoint i64* %T3 to i64
  %offsetcalc = mul i64 %plus1, 8
  %pmath5 = add i64 %intcast4, %offsetcalc
  %pointercast6 = inttoptr i64 %pmath5 to i64*
  %rootload = load i64, i64* %pointercast6
  %offsetcalc7 = mul i64 %glr, 8
  %pmath8 = add i64 %rootload, %offsetcalc7
  %pointercast9 = inttoptr i64 %pmath8 to i64*
  %deref = load i64, i64* %pointercast9
  %derefpntr = inttoptr i64 %deref to i64*
  %fpaccess = bitcast i64* %derefpntr to double*
  %fpret = load double, double* %fpaccess
  store double %fpret, double* %max
  %i = alloca i64
  store i64 1, i64* %i
  br label %while

while:                                            ; preds = %merge, %entry
  %i48 = load i64, i64* %i
  %da49 = load i64, i64* %da
  %ftmp = sitofp i64 %da49 to double
  %ftmp50 = sitofp i64 %i48 to double
  %tmp51 = fcmp olt double %ftmp50, %ftmp
  %booltmp52 = icmp ne i1 %tmp51, false
  br i1 %booltmp52, label %while_body, label %merge53

while_body:                                       ; preds = %while
  %i10 = load i64, i64* %i
  %free_alloc = alloca i64
  store i64 %i10, i64* %free_alloc
  %T11 = load i64*, i64** %T1
  %glr12 = load i64, i64* %free_alloc
  %offsetload13 = load i64, i64* %T11
  %plus114 = add i64 %offsetload13, 1
  %intcast15 = ptrtoint i64* %T11 to i64
  %offsetcalc16 = mul i64 %plus114, 8
  %pmath17 = add i64 %intcast15, %offsetcalc16
  %pointercast18 = inttoptr i64 %pmath17 to i64*
  %rootload19 = load i64, i64* %pointercast18
  %offsetcalc20 = mul i64 %glr12, 8
  %pmath21 = add i64 %rootload19, %offsetcalc20
  %pointercast22 = inttoptr i64 %pmath21 to i64*
  %deref23 = load i64, i64* %pointercast22
  %derefpntr24 = inttoptr i64 %deref23 to i64*
  %fpaccess25 = bitcast i64* %derefpntr24 to double*
  %fpret26 = load double, double* %fpaccess25
  %max27 = load double, double* %max
  %tmp = fcmp ogt double %fpret26, %max27
  %booltmp = icmp ne i1 %tmp, false
  br i1 %booltmp, label %then, label %else

merge:                                            ; preds = %else, %then
```

```
  %i46 = load i64, i64* %i
  %tmp47 = add i64 %i46, 1
  store i64 %tmp47, i64* %i
  br label %while

then:                                             ; preds = %while_body
  %i28 = load i64, i64* %i
  %free_alloc29 = alloca i64
  store i64 %i28, i64* %free_alloc29
  %T30 = load i64*, i64** %T1
  %glr31 = load i64, i64* %free_alloc29
  %offsetload32 = load i64, i64* %T30
  %plus133 = add i64 %offsetload32, 1
  %intcast34 = ptrtoint i64* %T30 to i64
  %offsetcalc35 = mul i64 %plus133, 8
  %pmath36 = add i64 %intcast34, %offsetcalc35
  %pointercast37 = inttoptr i64 %pmath36 to i64*
  %rootload38 = load i64, i64* %pointercast37
  %offsetcalc39 = mul i64 %glr31, 8
  %pmath40 = add i64 %rootload38, %offsetcalc39
  %pointercast41 = inttoptr i64 %pmath40 to i64*
  %deref42 = load i64, i64* %pointercast41
  %derefpntr43 = inttoptr i64 %deref42 to i64*
  %fpaccess44 = bitcast i64* %derefpntr43 to double*
  %fpret45 = load double, double* %fpaccess44
  store double %fpret45, double* %max
  br label %merge

else:                                             ; preds = %while_body
  br label %merge

merge53:                                          ; preds = %while
  %max54 = load double, double* %max
  ret double %max54
}

define i64 @maxInd(i64* %T) {
entry:
  %T1 = alloca i64*
  store i64* %T, i64** %T1
  %da = alloca i64
  %T2 = load i64*, i64** %T1
  %intcast = ptrtoint i64* %T2 to i64
  %pmath = add i64 %intcast, 8
  %pointercast = inttoptr i64 %pmath to i64*
  %tenresult = load i64, i64* %pointercast
  store i64 %tenresult, i64* %da
  %max = alloca i64
  store i64 0, i64* %max
  %i = alloca i64
  store i64 1, i64* %i
  br label %while

while:                                            ; preds = %merge, %entry
  %i32 = load i64, i64* %i
  %da33 = load i64, i64* %da
```

```
  %ftmp = sitofp i64 %da33 to double
  %ftmp34 = sitofp i64 %i32 to double
  %tmp35 = fcmp olt double %ftmp34, %ftmp
  %booltmp36 = icmp ne i1 %tmp35, false
  br i1 %booltmp36, label %while_body, label %merge37

while_body:                                       ; preds = %while
  %i3 = load i64, i64* %i
  %free_alloc = alloca i64
  store i64 %i3, i64* %free_alloc
  %T4 = load i64*, i64** %T1
  %glr = load i64, i64* %free_alloc
  %offsetload = load i64, i64* %T4
  %plus1 = add i64 %offsetload, 1
  %intcast5 = ptrtoint i64* %T4 to i64
  %offsetcalc = mul i64 %plus1, 8
  %pmath6 = add i64 %intcast5, %offsetcalc
  %pointercast7 = inttoptr i64 %pmath6 to i64*
  %rootload = load i64, i64* %pointercast7
  %offsetcalc8 = mul i64 %glr, 8
  %pmath9 = add i64 %rootload, %offsetcalc8
  %pointercast10 = inttoptr i64 %pmath9 to i64*
  %deref = load i64, i64* %pointercast10
  %derefpntr = inttoptr i64 %deref to i64*
  %fpaccess = bitcast i64* %derefpntr to double*
  %fpret = load double, double* %fpaccess
  %max11 = load i64, i64* %max
  %free_alloc12 = alloca i64
  store i64 %max11, i64* %free_alloc12
  %T13 = load i64*, i64** %T1
  %glr14 = load i64, i64* %free_alloc12
  %offsetload15 = load i64, i64* %T13
  %plus116 = add i64 %offsetload15, 1
  %intcast17 = ptrtoint i64* %T13 to i64
  %offsetcalc18 = mul i64 %plus116, 8
  %pmath19 = add i64 %intcast17, %offsetcalc18
  %pointercast20 = inttoptr i64 %pmath19 to i64*
  %rootload21 = load i64, i64* %pointercast20
  %offsetcalc22 = mul i64 %glr14, 8
  %pmath23 = add i64 %rootload21, %offsetcalc22
  %pointercast24 = inttoptr i64 %pmath23 to i64*
  %deref25 = load i64, i64* %pointercast24
  %derefpntr26 = inttoptr i64 %deref25 to i64*
  %fpaccess27 = bitcast i64* %derefpntr26 to double*
  %fpret28 = load double, double* %fpaccess27
  %tmp = fcmp ogt double %fpret, %fpret28
  %booltmp = icmp ne i1 %tmp, false
  br i1 %booltmp, label %then, label %else

merge:                                            ; preds = %else, %then
  %i30 = load i64, i64* %i
  %tmp31 = add i64 %i30, 1
  store i64 %tmp31, i64* %i
  br label %while

then:                                             ; preds = %while_body
```

```
  %i29 = load i64, i64* %i
  store i64 %i29, i64* %max
  br label %merge

else:                                             ; preds = %while_body
  br label %merge

merge37:                                          ; preds = %while
  %max38 = load i64, i64* %max
  ret i64 %max38
}

define double @abs(double %n) {
entry:
  %n1 = alloca double
  store double %n, double* %n1
  %n2 = load double, double* %n1
  %tmp = fcmp olt double %n2, 0.000000e+00
  %booltmp = icmp ne i1 %tmp, false
  br i1 %booltmp, label %then, label %else

merge:                                            ; preds = %else
  %n5 = load double, double* %n1
  ret double %n5

then:                                             ; preds = %entry
  %n3 = load double, double* %n1
  %tmp4 = fsub double 0.000000e+00, %n3
  ret double %tmp4

else:                                             ; preds = %entry
  br label %merge
}

declare noalias i8* @malloc(i32)
```

7.1.3 Example: Perceptron Demo

The third demo implements a simple machine learning algorithm, perceptron, which finds a linear classifier that best separates the given data points. This example uses a four-dimensional boolean function as the data.
The LaTenS source code:

```
%% Perceptron
int main() {
   let trainingData = [[0, 0, 0, 0, 1],
                       [0, 0, 0, 1, 1],
                       [0, 0, 1, 0, 1],
                       [0, 0, 1, 1, 1],
                       [0, 1, 0, 0, 1],
                       [0, 1, 0, 1, 1],
                       [0, 1, 1, 0, 1],
                       [0, 1, 1, 1, 1],
                       [1, 0, 0, 0, 1],
                       [1, 0, 0, 1, 1],
```

```
                        [1, 0, 1, 0, 1],
                        [1, 0, 1, 1, 1],
                        [1, 1, 0, 0, 1],
                        [1, 1, 0, 1, 1],
                        [1, 1, 1, 0, 1],
                        [1, 1, 1, 1, 1]];
    let trainingLabels = [-1, -1, -1, 1, -1, -1, -1, 1, -1, -1, -1, 1, 1, 1, 1, 1];


    let theta_{dim(trainingData, 1)};

    let check = 1; % check if the classifier makes any mistake
    let count = 0; % returns the number of iterations needed to perfectly classify linearly
        separable data (or how many iterations we tried)
    let mistakes = 0;

    %% Halt after 500 iterations over the dataset
    while (check == 1 && count < 500) {
       check = 0;
       count = count + 1;

       %% Loop through the entire dataset
       for (let i = 0; i < dim(trainingLabels, 0); i = i + 1) {

          let y = trainingLabels_{i};
          let x = trainingData_{i,a};
          %% the current row of the matrix dotted with theta
          let guess = x_{a} * theta_{a};

          %% If the classifier has made a mistake, update it
          if (guess * y <= 0)
          %% Update the classifier
          {
             check = 1;
             mistakes = mistakes + 1;
             %% For each element of theta, add y*x_i
             for (let j = 0; j < dim(theta, 0); j = j + 1) {
                theta_{j} = theta_{j} + (y * x_{j});
             }
          }
       }
    }

    print("FINAL CLASSIFIER:\n");
    print(theta);
    print(itos(mistakes) ^ " mistakes made over " ^ itos(count) ^ " iterations.\n");

    return 0;
}
```

And the LLVM IR:

```
; ModuleID = 'LaTenS'

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
```

```
@fmt3 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt = private unnamed_addr constant [5 x i8] c"%.6f\00"
@"FINAL CLASSIFIER:\0A" = private unnamed_addr constant [19 x i8] c"FINAL CLASSIFIER:\0A\00"
@" mistakes made over " = private unnamed_addr constant [21 x i8] c" mistakes made over \00"
@" iterations.\0A" = private unnamed_addr constant [14 x i8] c" iterations.\0A\00"
@fmt.1 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.2 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.3 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.4 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.5 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.6 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.7 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.8 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.9 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.10 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.11 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.12 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.13 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.14 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.15 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.16 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.17 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.18 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.19 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.20 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.21 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.22 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.23 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.24 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.25 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.26 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.27 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.28 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.29 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.30 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.31 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.32 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.33 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.34 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.35 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.36 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.37 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.38 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.39 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.40 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.41 = private unnamed_addr constant [2 x i8] c"\0A\00"
@ftosfmt.42 = private unnamed_addr constant [5 x i8] c"%.6f\00"
@fmt.43 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt2.44 = private unnamed_addr constant [6 x i8] c"%.6f\0A\00"
@fmt3.45 = private unnamed_addr constant [7 x i8] c"%.6f, \00"
@concatfmt.46 = private unnamed_addr constant [5 x i8] c"%s%s\00"
@itosfmt.47 = private unnamed_addr constant [3 x i8] c"%d\00"
@nlstr.48 = private unnamed_addr constant [2 x i8] c"\0A\00"
```

```
@ftosfmt.49 = private unnamed_addr constant [5 x i8] c"%.6f\00"

declare i64 @printf(i8*, ...)

declare i32 @strlen(i8*)

declare i64 @sprintf(i8*, i8*, ...)

declare i64 @atoi(i8*)

declare double @atof(i8*)

declare i64 @fopen(i8*, i8*)

declare i64 @fclose(i64)

declare i32 @fread(i64*, i32, i32, i64*)

declare i64 @fwrite(i64*, i32, i32, i64*)

declare double @sqrt(double)

define i64 @main() {
entry:
  %trainingData = alloca i64*
  %0 = trunc i64 4 to i32
  %mallocsize = mul i32 %0, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall = tail call i8* @malloc(i32 %mallocsize)
  %immediate_tmp = bitcast i8* %malloccall to i64*
  store i64 2, i64* %immediate_tmp
  %intcast = ptrtoint i64* %immediate_tmp to i64
  %pmath = add i64 %intcast, 8
  %pointercast = inttoptr i64 %pmath to i64*
  store i64 16, i64* %pointercast
  %intcast1 = ptrtoint i64* %immediate_tmp to i64
  %pmath2 = add i64 %intcast1, 16
  %pointercast3 = inttoptr i64 %pmath2 to i64*
  store i64 5, i64* %pointercast3
  %intcast4 = ptrtoint i64* %immediate_tmp to i64
  %pmath5 = add i64 %intcast4, 24
  %pointercast6 = inttoptr i64 %pmath5 to i64*
  %1 = trunc i64 16 to i32
  %mallocsize7 = mul i32 %1, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall8 = tail call i8* @malloc(i32 %mallocsize7)
  %fstarstorage = bitcast i8* %malloccall8 to i64*
  %2 = trunc i64 5 to i32
  %mallocsize9 = mul i32 %2, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall10 = tail call i8* @malloc(i32 %mallocsize9)
  %fstorage = bitcast i8* %malloccall10 to i64*
  %intcast11 = ptrtoint i64* %fstorage to i64
  %pmath12 = add i64 %intcast11, 0
  %pointercast13 = inttoptr i64 %pmath12 to i64*
  %malloccall14 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
      i32 1) to i32))
  %factual = bitcast i8* %malloccall14 to double*
  %ptwtointforfpstorage = ptrtoint double* %factual to i64
```

```
    store double 0.000000e+00, double* %factual
    store i64 %ptwtointforfpstorage, i64* %pointercast13
    %intcast15 = ptrtoint i64* %fstorage to i64
    %pmath16 = add i64 %intcast15, 8
    %pointercast17 = inttoptr i64 %pmath16 to i64*
    %malloccall18 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
        i32 1) to i32))
    %factual19 = bitcast i8* %malloccall18 to double*
    %ptwtointforfpstorage20 = ptrtoint double* %factual19 to i64
    store double 0.000000e+00, double* %factual19
    store i64 %ptwtointforfpstorage20, i64* %pointercast17
    %intcast21 = ptrtoint i64* %fstorage to i64
    %pmath22 = add i64 %intcast21, 16
    %pointercast23 = inttoptr i64 %pmath22 to i64*
    %malloccall24 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
        i32 1) to i32))
    %factual25 = bitcast i8* %malloccall24 to double*
    %ptwtointforfpstorage26 = ptrtoint double* %factual25 to i64
    store double 0.000000e+00, double* %factual25
    store i64 %ptwtointforfpstorage26, i64* %pointercast23
    %intcast27 = ptrtoint i64* %fstorage to i64
    %pmath28 = add i64 %intcast27, 24
    %pointercast29 = inttoptr i64 %pmath28 to i64*
    %malloccall30 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
        i32 1) to i32))
    %factual31 = bitcast i8* %malloccall30 to double*
    %ptwtointforfpstorage32 = ptrtoint double* %factual31 to i64
    store double 0.000000e+00, double* %factual31
    store i64 %ptwtointforfpstorage32, i64* %pointercast29
    %intcast33 = ptrtoint i64* %fstorage to i64
    %pmath34 = add i64 %intcast33, 32
    %pointercast35 = inttoptr i64 %pmath34 to i64*
    %malloccall36 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
        i32 1) to i32))
    %factual37 = bitcast i8* %malloccall36 to double*
    %ptwtointforfpstorage38 = ptrtoint double* %factual37 to i64
    store double 1.000000e+00, double* %factual37
    store i64 %ptwtointforfpstorage38, i64* %pointercast35
    %ptrtointfornlstorage = ptrtoint i64* %fstorage to i64
    store i64 %ptrtointfornlstorage, i64* %fstarstorage
    %ptwtointfornlstorage = ptrtoint i64* %fstarstorage to i64
    %nextptr = add i64 %ptwtointfornlstorage, 8
    %nextptrptr = inttoptr i64 %nextptr to i64*
    %3 = trunc i64 5 to i32
    %mallocsize39 = mul i32 %3, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
    %malloccall40 = tail call i8* @malloc(i32 %mallocsize39)
    %fstorage41 = bitcast i8* %malloccall40 to i64*
    %intcast42 = ptrtoint i64* %fstorage41 to i64
    %pmath43 = add i64 %intcast42, 0
    %pointercast44 = inttoptr i64 %pmath43 to i64*
    %malloccall45 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
        i32 1) to i32))
    %factual46 = bitcast i8* %malloccall45 to double*
    %ptwtointforfpstorage47 = ptrtoint double* %factual46 to i64
    store double 0.000000e+00, double* %factual46
    store i64 %ptwtointforfpstorage47, i64* %pointercast44
```

```
  %intcast48 = ptrtoint i64* %fstorage41 to i64
  %pmath49 = add i64 %intcast48, 8
  %pointercast50 = inttoptr i64 %pmath49 to i64*
  %malloccall51 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
      i32 1) to i32))
  %factual52 = bitcast i8* %malloccall51 to double*
  %ptwtointforfpstorage53 = ptrtoint double* %factual52 to i64
  store double 0.000000e+00, double* %factual52
  store i64 %ptwtointforfpstorage53, i64* %pointercast50
  %intcast54 = ptrtoint i64* %fstorage41 to i64
  %pmath55 = add i64 %intcast54, 16
  %pointercast56 = inttoptr i64 %pmath55 to i64*
  %malloccall57 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
      i32 1) to i32))
  %factual58 = bitcast i8* %malloccall57 to double*
  %ptwtointforfpstorage59 = ptrtoint double* %factual58 to i64
  store double 0.000000e+00, double* %factual58
  store i64 %ptwtointforfpstorage59, i64* %pointercast56
  %intcast60 = ptrtoint i64* %fstorage41 to i64
  %pmath61 = add i64 %intcast60, 24
  %pointercast62 = inttoptr i64 %pmath61 to i64*
  %malloccall63 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
      i32 1) to i32))
  %factual64 = bitcast i8* %malloccall63 to double*
  %ptwtointforfpstorage65 = ptrtoint double* %factual64 to i64
  store double 1.000000e+00, double* %factual64
  store i64 %ptwtointforfpstorage65, i64* %pointercast62
  %intcast66 = ptrtoint i64* %fstorage41 to i64
  %pmath67 = add i64 %intcast66, 32
  %pointercast68 = inttoptr i64 %pmath67 to i64*
  %malloccall69 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
      i32 1) to i32))
  %factual70 = bitcast i8* %malloccall69 to double*
  %ptwtointforfpstorage71 = ptrtoint double* %factual70 to i64
  store double 1.000000e+00, double* %factual70
  store i64 %ptwtointforfpstorage71, i64* %pointercast68
  %ptrtointfornlstorage72 = ptrtoint i64* %fstorage41 to i64
  store i64 %ptrtointfornlstorage72, i64* %nextptrptr
  %ptwtointfornlstorage73 = ptrtoint i64* %nextptrptr to i64
  %nextptr74 = add i64 %ptwtointfornlstorage73, 8
  %nextptrptr75 = inttoptr i64 %nextptr74 to i64*
  %4 = trunc i64 5 to i32
  %mallocsize76 = mul i32 %4, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall77 = tail call i8* @malloc(i32 %mallocsize76)
  %fstorage78 = bitcast i8* %malloccall77 to i64*
  %intcast79 = ptrtoint i64* %fstorage78 to i64
  %pmath80 = add i64 %intcast79, 0
  %pointercast81 = inttoptr i64 %pmath80 to i64*
  %malloccall82 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
      i32 1) to i32))
  %factual83 = bitcast i8* %malloccall82 to double*
  %ptwtointforfpstorage84 = ptrtoint double* %factual83 to i64
  store double 0.000000e+00, double* %factual83
  store i64 %ptwtointforfpstorage84, i64* %pointercast81
  %intcast85 = ptrtoint i64* %fstorage78 to i64
  %pmath86 = add i64 %intcast85, 8
```

```
  %pointercast87 = inttoptr i64 %pmath86 to i64*
  %malloccall88 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
      i32 1) to i32))
  %factual89 = bitcast i8* %malloccall88 to double*
  %ptwtointforfpstorage90 = ptrtoint double* %factual89 to i64
  store double 0.000000e+00, double* %factual89
  store i64 %ptwtointforfpstorage90, i64* %pointercast87
  %intcast91 = ptrtoint i64* %fstorage78 to i64
  %pmath92 = add i64 %intcast91, 16
  %pointercast93 = inttoptr i64 %pmath92 to i64*
  %malloccall94 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null,
      i32 1) to i32))
  %factual95 = bitcast i8* %malloccall94 to double*
  %ptwtointforfpstorage96 = ptrtoint double* %factual95 to i64
  store double 1.000000e+00, double* %factual95
  store i64 %ptwtointforfpstorage96, i64* %pointercast93
  %intcast97 = ptrtoint i64* %fstorage78 to i64
  %pmath98 = add i64 %intcast97, 24
  %pointercast99 = inttoptr i64 %pmath98 to i64*
  %malloccall100 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual101 = bitcast i8* %malloccall100 to double*
  %ptwtointforfpstorage102 = ptrtoint double* %factual101 to i64
  store double 0.000000e+00, double* %factual101
  store i64 %ptwtointforfpstorage102, i64* %pointercast99
  %intcast103 = ptrtoint i64* %fstorage78 to i64
  %pmath104 = add i64 %intcast103, 32
  %pointercast105 = inttoptr i64 %pmath104 to i64*
  %malloccall106 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual107 = bitcast i8* %malloccall106 to double*
  %ptwtointforfpstorage108 = ptrtoint double* %factual107 to i64
  store double 1.000000e+00, double* %factual107
  store i64 %ptwtointforfpstorage108, i64* %pointercast105
  %ptrtointfornlstorage109 = ptrtoint i64* %fstorage78 to i64
  store i64 %ptrtointfornlstorage109, i64* %nextptrptr75
  %ptwtointfornlstorage110 = ptrtoint i64* %nextptrptr75 to i64
  %nextptr111 = add i64 %ptwtointfornlstorage110, 8
  %nextptrptr112 = inttoptr i64 %nextptr111 to i64*
  %5 = trunc i64 5 to i32
  %mallocsize113 = mul i32 %5, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall114 = tail call i8* @malloc(i32 %mallocsize113)
  %fstorage115 = bitcast i8* %malloccall114 to i64*
  %intcast116 = ptrtoint i64* %fstorage115 to i64
  %pmath117 = add i64 %intcast116, 0
  %pointercast118 = inttoptr i64 %pmath117 to i64*
  %malloccall119 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual120 = bitcast i8* %malloccall119 to double*
  %ptwtointforfpstorage121 = ptrtoint double* %factual120 to i64
  store double 0.000000e+00, double* %factual120
  store i64 %ptwtointforfpstorage121, i64* %pointercast118
  %intcast122 = ptrtoint i64* %fstorage115 to i64
  %pmath123 = add i64 %intcast122, 8
  %pointercast124 = inttoptr i64 %pmath123 to i64*
  %malloccall125 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
```

```
        null, i32 1) to i32))
  %factual126 = bitcast i8* %malloccall125 to double*
  %ptwtointforfpstorage127 = ptrtoint double* %factual126 to i64
  store double 0.000000e+00, double* %factual126
  store i64 %ptwtointforfpstorage127, i64* %pointercast124
  %intcast128 = ptrtoint i64* %fstorage115 to i64
  %pmath129 = add i64 %intcast128, 16
  %pointercast130 = inttoptr i64 %pmath129 to i64*
  %malloccall131 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual132 = bitcast i8* %malloccall131 to double*
  %ptwtointforfpstorage133 = ptrtoint double* %factual132 to i64
  store double 1.000000e+00, double* %factual132
  store i64 %ptwtointforfpstorage133, i64* %pointercast130
  %intcast134 = ptrtoint i64* %fstorage115 to i64
  %pmath135 = add i64 %intcast134, 24
  %pointercast136 = inttoptr i64 %pmath135 to i64*
  %malloccall137 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual138 = bitcast i8* %malloccall137 to double*
  %ptwtointforfpstorage139 = ptrtoint double* %factual138 to i64
  store double 1.000000e+00, double* %factual138
  store i64 %ptwtointforfpstorage139, i64* %pointercast136
  %intcast140 = ptrtoint i64* %fstorage115 to i64
  %pmath141 = add i64 %intcast140, 32
  %pointercast142 = inttoptr i64 %pmath141 to i64*
  %malloccall143 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual144 = bitcast i8* %malloccall143 to double*
  %ptwtointforfpstorage145 = ptrtoint double* %factual144 to i64
  store double 1.000000e+00, double* %factual144
  store i64 %ptwtointforfpstorage145, i64* %pointercast142
  %ptrtointfornlstorage146 = ptrtoint i64* %fstorage115 to i64
  store i64 %ptrtointfornlstorage146, i64* %nextptrptr112
  %ptwtointfornlstorage147 = ptrtoint i64* %nextptrptr112 to i64
  %nextptr148 = add i64 %ptwtointfornlstorage147, 8
  %nextptrptr149 = inttoptr i64 %nextptr148 to i64*
  %6 = trunc i64 5 to i32
  %mallocsize150 = mul i32 %6, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall151 = tail call i8* @malloc(i32 %mallocsize150)
  %fstorage152 = bitcast i8* %malloccall151 to i64*
  %intcast153 = ptrtoint i64* %fstorage152 to i64
  %pmath154 = add i64 %intcast153, 0
  %pointercast155 = inttoptr i64 %pmath154 to i64*
  %malloccall156 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual157 = bitcast i8* %malloccall156 to double*
  %ptwtointforfpstorage158 = ptrtoint double* %factual157 to i64
  store double 0.000000e+00, double* %factual157
  store i64 %ptwtointforfpstorage158, i64* %pointercast155
  %intcast159 = ptrtoint i64* %fstorage152 to i64
  %pmath160 = add i64 %intcast159, 8
  %pointercast161 = inttoptr i64 %pmath160 to i64*
  %malloccall162 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual163 = bitcast i8* %malloccall162 to double*
```

```
  %ptwtointforfpstorage164 = ptrtoint double* %factual163 to i64
  store double 1.000000e+00, double* %factual163
  store i64 %ptwtointforfpstorage164, i64* %pointercast161
  %intcast165 = ptrtoint i64* %fstorage152 to i64
  %pmath166 = add i64 %intcast165, 16
  %pointercast167 = inttoptr i64 %pmath166 to i64*
  %malloccall168 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual169 = bitcast i8* %malloccall168 to double*
  %ptwtointforfpstorage170 = ptrtoint double* %factual169 to i64
  store double 0.000000e+00, double* %factual169
  store i64 %ptwtointforfpstorage170, i64* %pointercast167
  %intcast171 = ptrtoint i64* %fstorage152 to i64
  %pmath172 = add i64 %intcast171, 24
  %pointercast173 = inttoptr i64 %pmath172 to i64*
  %malloccall174 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual175 = bitcast i8* %malloccall174 to double*
  %ptwtointforfpstorage176 = ptrtoint double* %factual175 to i64
  store double 0.000000e+00, double* %factual175
  store i64 %ptwtointforfpstorage176, i64* %pointercast173
  %intcast177 = ptrtoint i64* %fstorage152 to i64
  %pmath178 = add i64 %intcast177, 32
  %pointercast179 = inttoptr i64 %pmath178 to i64*
  %malloccall180 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual181 = bitcast i8* %malloccall180 to double*
  %ptwtointforfpstorage182 = ptrtoint double* %factual181 to i64
  store double 1.000000e+00, double* %factual181
  store i64 %ptwtointforfpstorage182, i64* %pointercast179
  %ptrtointfornlstorage183 = ptrtoint i64* %fstorage152 to i64
  store i64 %ptrtointfornlstorage183, i64* %nextptrptr149
  %ptwtointfornlstorage184 = ptrtoint i64* %nextptrptr149 to i64
  %nextptr185 = add i64 %ptwtointfornlstorage184, 8
  %nextptrptr186 = inttoptr i64 %nextptr185 to i64*
  %7 = trunc i64 5 to i32
  %mallocsize187 = mul i32 %7, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall188 = tail call i8* @malloc(i32 %mallocsize187)
  %fstorage189 = bitcast i8* %malloccall188 to i64*
  %intcast190 = ptrtoint i64* %fstorage189 to i64
  %pmath191 = add i64 %intcast190, 0
  %pointercast192 = inttoptr i64 %pmath191 to i64*
  %malloccall193 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual194 = bitcast i8* %malloccall193 to double*
  %ptwtointforfpstorage195 = ptrtoint double* %factual194 to i64
  store double 0.000000e+00, double* %factual194
  store i64 %ptwtointforfpstorage195, i64* %pointercast192
  %intcast196 = ptrtoint i64* %fstorage189 to i64
  %pmath197 = add i64 %intcast196, 8
  %pointercast198 = inttoptr i64 %pmath197 to i64*
  %malloccall199 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual200 = bitcast i8* %malloccall199 to double*
  %ptwtointforfpstorage201 = ptrtoint double* %factual200 to i64
  store double 1.000000e+00, double* %factual200
```

```
370   store i64 %ptwtointforfpstorage201, i64* %pointercast198
371   %intcast202 = ptrtoint i64* %fstorage189 to i64
372   %pmath203 = add i64 %intcast202, 16
373   %pointercast204 = inttoptr i64 %pmath203 to i64*
374   %malloccall205 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
375   %factual206 = bitcast i8* %malloccall205 to double*
376   %ptwtointforfpstorage207 = ptrtoint double* %factual206 to i64
377   store double 0.000000e+00, double* %factual206
378   store i64 %ptwtointforfpstorage207, i64* %pointercast204
379   %intcast208 = ptrtoint i64* %fstorage189 to i64
380   %pmath209 = add i64 %intcast208, 24
381   %pointercast210 = inttoptr i64 %pmath209 to i64*
382   %malloccall211 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
383   %factual212 = bitcast i8* %malloccall211 to double*
384   %ptwtointforfpstorage213 = ptrtoint double* %factual212 to i64
385   store double 1.000000e+00, double* %factual212
386   store i64 %ptwtointforfpstorage213, i64* %pointercast210
387   %intcast214 = ptrtoint i64* %fstorage189 to i64
388   %pmath215 = add i64 %intcast214, 32
389   %pointercast216 = inttoptr i64 %pmath215 to i64*
390   %malloccall217 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
391   %factual218 = bitcast i8* %malloccall217 to double*
392   %ptwtointforfpstorage219 = ptrtoint double* %factual218 to i64
393   store double 1.000000e+00, double* %factual218
394   store i64 %ptwtointforfpstorage219, i64* %pointercast216
395   %ptrtointfornlstorage220 = ptrtoint i64* %fstorage189 to i64
396   store i64 %ptrtointfornlstorage220, i64* %nextptrptr186
397   %ptwtointfornlstorage221 = ptrtoint i64* %nextptrptr186 to i64
398   %nextptr222 = add i64 %ptwtointfornlstorage221, 8
399   %nextptrptr223 = inttoptr i64 %nextptr222 to i64*
400   %8 = trunc i64 5 to i32
401   %mallocsize224 = mul i32 %8, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
402   %malloccall225 = tail call i8* @malloc(i32 %mallocsize224)
403   %fstorage226 = bitcast i8* %malloccall225 to i64*
404   %intcast227 = ptrtoint i64* %fstorage226 to i64
405   %pmath228 = add i64 %intcast227, 0
406   %pointercast229 = inttoptr i64 %pmath228 to i64*
407   %malloccall230 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
408   %factual231 = bitcast i8* %malloccall230 to double*
409   %ptwtointforfpstorage232 = ptrtoint double* %factual231 to i64
410   store double 0.000000e+00, double* %factual231
411   store i64 %ptwtointforfpstorage232, i64* %pointercast229
412   %intcast233 = ptrtoint i64* %fstorage226 to i64
413   %pmath234 = add i64 %intcast233, 8
414   %pointercast235 = inttoptr i64 %pmath234 to i64*
415   %malloccall236 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
416   %factual237 = bitcast i8* %malloccall236 to double*
417   %ptwtointforfpstorage238 = ptrtoint double* %factual237 to i64
418   store double 1.000000e+00, double* %factual237
419   store i64 %ptwtointforfpstorage238, i64* %pointercast235
420   %intcast239 = ptrtoint i64* %fstorage226 to i64
```

```
  %pmath240 = add i64 %intcast239, 16
  %pointercast241 = inttoptr i64 %pmath240 to i64*
  %malloccall242 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual243 = bitcast i8* %malloccall242 to double*
  %ptwtointforfpstorage244 = ptrtoint double* %factual243 to i64
  store double 1.000000e+00, double* %factual243
  store i64 %ptwtointforfpstorage244, i64* %pointercast241
  %intcast245 = ptrtoint i64* %fstorage226 to i64
  %pmath246 = add i64 %intcast245, 24
  %pointercast247 = inttoptr i64 %pmath246 to i64*
  %malloccall248 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual249 = bitcast i8* %malloccall248 to double*
  %ptwtointforfpstorage250 = ptrtoint double* %factual249 to i64
  store double 0.000000e+00, double* %factual249
  store i64 %ptwtointforfpstorage250, i64* %pointercast247
  %intcast251 = ptrtoint i64* %fstorage226 to i64
  %pmath252 = add i64 %intcast251, 32
  %pointercast253 = inttoptr i64 %pmath252 to i64*
  %malloccall254 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual255 = bitcast i8* %malloccall254 to double*
  %ptwtointforfpstorage256 = ptrtoint double* %factual255 to i64
  store double 1.000000e+00, double* %factual255
  store i64 %ptwtointforfpstorage256, i64* %pointercast253
  %ptrtointfornlstorage257 = ptrtoint i64* %fstorage226 to i64
  store i64 %ptrtointfornlstorage257, i64* %nextptrptr223
  %ptwtointfornlstorage258 = ptrtoint i64* %nextptrptr223 to i64
  %nextptr259 = add i64 %ptwtointfornlstorage258, 8
  %nextptrptr260 = inttoptr i64 %nextptr259 to i64*
  %9 = trunc i64 5 to i32
  %mallocsize261 = mul i32 %9, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall262 = tail call i8* @malloc(i32 %mallocsize261)
  %fstorage263 = bitcast i8* %malloccall262 to i64*
  %intcast264 = ptrtoint i64* %fstorage263 to i64
  %pmath265 = add i64 %intcast264, 0
  %pointercast266 = inttoptr i64 %pmath265 to i64*
  %malloccall267 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual268 = bitcast i8* %malloccall267 to double*
  %ptwtointforfpstorage269 = ptrtoint double* %factual268 to i64
  store double 0.000000e+00, double* %factual268
  store i64 %ptwtointforfpstorage269, i64* %pointercast266
  %intcast270 = ptrtoint i64* %fstorage263 to i64
  %pmath271 = add i64 %intcast270, 8
  %pointercast272 = inttoptr i64 %pmath271 to i64*
  %malloccall273 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual274 = bitcast i8* %malloccall273 to double*
  %ptwtointforfpstorage275 = ptrtoint double* %factual274 to i64
  store double 1.000000e+00, double* %factual274
  store i64 %ptwtointforfpstorage275, i64* %pointercast272
  %intcast276 = ptrtoint i64* %fstorage263 to i64
  %pmath277 = add i64 %intcast276, 16
  %pointercast278 = inttoptr i64 %pmath277 to i64*
```

```
%malloccall279 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null, i32 1) to i32))
%factual280 = bitcast i8* %malloccall279 to double*
%ptwtointforfpstorage281 = ptrtoint double* %factual280 to i64
store double 1.000000e+00, double* %factual280
store i64 %ptwtointforfpstorage281, i64* %pointercast278
%intcast282 = ptrtoint i64* %fstorage263 to i64
%pmath283 = add i64 %intcast282, 24
%pointercast284 = inttoptr i64 %pmath283 to i64*
%malloccall285 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null, i32 1) to i32))
%factual286 = bitcast i8* %malloccall285 to double*
%ptwtointforfpstorage287 = ptrtoint double* %factual286 to i64
store double 1.000000e+00, double* %factual286
store i64 %ptwtointforfpstorage287, i64* %pointercast284
%intcast288 = ptrtoint i64* %fstorage263 to i64
%pmath289 = add i64 %intcast288, 32
%pointercast290 = inttoptr i64 %pmath289 to i64*
%malloccall291 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null, i32 1) to i32))
%factual292 = bitcast i8* %malloccall291 to double*
%ptwtointforfpstorage293 = ptrtoint double* %factual292 to i64
store double 1.000000e+00, double* %factual292
store i64 %ptwtointforfpstorage293, i64* %pointercast290
%ptrtointfornlstorage294 = ptrtoint i64* %fstorage263 to i64
store i64 %ptrtointfornlstorage294, i64* %nextptrptr260
%ptwtointfornlstorage295 = ptrtoint i64* %nextptrptr260 to i64
%nextptr296 = add i64 %ptwtointfornlstorage295, 8
%nextptrptr297 = inttoptr i64 %nextptr296 to i64*
%10 = trunc i64 5 to i32
%mallocsize298 = mul i32 %10, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
%malloccall299 = tail call i8* @malloc(i32 %mallocsize298)
%fstorage300 = bitcast i8* %malloccall299 to i64*
%intcast301 = ptrtoint i64* %fstorage300 to i64
%pmath302 = add i64 %intcast301, 0
%pointercast303 = inttoptr i64 %pmath302 to i64*
%malloccall304 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null, i32 1) to i32))
%factual305 = bitcast i8* %malloccall304 to double*
%ptwtointforfpstorage306 = ptrtoint double* %factual305 to i64
store double 1.000000e+00, double* %factual305
store i64 %ptwtointforfpstorage306, i64* %pointercast303
%intcast307 = ptrtoint i64* %fstorage300 to i64
%pmath308 = add i64 %intcast307, 8
%pointercast309 = inttoptr i64 %pmath308 to i64*
%malloccall310 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null, i32 1) to i32))
%factual311 = bitcast i8* %malloccall310 to double*
%ptwtointforfpstorage312 = ptrtoint double* %factual311 to i64
store double 0.000000e+00, double* %factual311
store i64 %ptwtointforfpstorage312, i64* %pointercast309
%intcast313 = ptrtoint i64* %fstorage300 to i64
%pmath314 = add i64 %intcast313, 16
%pointercast315 = inttoptr i64 %pmath314 to i64*
%malloccall316 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null, i32 1) to i32))
```

```
  %factual317 = bitcast i8* %malloccall316 to double*
  %ptwtointforfpstorage318 = ptrtoint double* %factual317 to i64
  store double 0.000000e+00, double* %factual317
  store i64 %ptwtointforfpstorage318, i64* %pointercast315
  %intcast319 = ptrtoint i64* %fstorage300 to i64
  %pmath320 = add i64 %intcast319, 24
  %pointercast321 = inttoptr i64 %pmath320 to i64*
  %malloccall322 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual323 = bitcast i8* %malloccall322 to double*
  %ptwtointforfpstorage324 = ptrtoint double* %factual323 to i64
  store double 0.000000e+00, double* %factual323
  store i64 %ptwtointforfpstorage324, i64* %pointercast321
  %intcast325 = ptrtoint i64* %fstorage300 to i64
  %pmath326 = add i64 %intcast325, 32
  %pointercast327 = inttoptr i64 %pmath326 to i64*
  %malloccall328 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual329 = bitcast i8* %malloccall328 to double*
  %ptwtointforfpstorage330 = ptrtoint double* %factual329 to i64
  store double 1.000000e+00, double* %factual329
  store i64 %ptwtointforfpstorage330, i64* %pointercast327
  %ptrtointfornlstorage331 = ptrtoint i64* %fstorage300 to i64
  store i64 %ptrtointfornlstorage331, i64* %nextptrptr297
  %ptwtointfornlstorage332 = ptrtoint i64* %nextptrptr297 to i64
  %nextptr333 = add i64 %ptwtointfornlstorage332, 8
  %nextptrptr334 = inttoptr i64 %nextptr333 to i64*
  %11 = trunc i64 5 to i32
  %mallocsize335 = mul i32 %11, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall336 = tail call i8* @malloc(i32 %mallocsize335)
  %fstorage337 = bitcast i8* %malloccall336 to i64*
  %intcast338 = ptrtoint i64* %fstorage337 to i64
  %pmath339 = add i64 %intcast338, 0
  %pointercast340 = inttoptr i64 %pmath339 to i64*
  %malloccall341 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual342 = bitcast i8* %malloccall341 to double*
  %ptwtointforfpstorage343 = ptrtoint double* %factual342 to i64
  store double 1.000000e+00, double* %factual342
  store i64 %ptwtointforfpstorage343, i64* %pointercast340
  %intcast344 = ptrtoint i64* %fstorage337 to i64
  %pmath345 = add i64 %intcast344, 8
  %pointercast346 = inttoptr i64 %pmath345 to i64*
  %malloccall347 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual348 = bitcast i8* %malloccall347 to double*
  %ptwtointforfpstorage349 = ptrtoint double* %factual348 to i64
  store double 0.000000e+00, double* %factual348
  store i64 %ptwtointforfpstorage349, i64* %pointercast346
  %intcast350 = ptrtoint i64* %fstorage337 to i64
  %pmath351 = add i64 %intcast350, 16
  %pointercast352 = inttoptr i64 %pmath351 to i64*
  %malloccall353 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual354 = bitcast i8* %malloccall353 to double*
  %ptwtointforfpstorage355 = ptrtoint double* %factual354 to i64
```

```
573   store double 0.000000e+00, double* %factual354
574   store i64 %ptwtointforfpstorage355, i64* %pointercast352
575   %intcast356 = ptrtoint i64* %fstorage337 to i64
576   %pmath357 = add i64 %intcast356, 24
577   %pointercast358 = inttoptr i64 %pmath357 to i64*
578   %malloccall359 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
579   %factual360 = bitcast i8* %malloccall359 to double*
580   %ptwtointforfpstorage361 = ptrtoint double* %factual360 to i64
581   store double 1.000000e+00, double* %factual360
582   store i64 %ptwtointforfpstorage361, i64* %pointercast358
583   %intcast362 = ptrtoint i64* %fstorage337 to i64
584   %pmath363 = add i64 %intcast362, 32
585   %pointercast364 = inttoptr i64 %pmath363 to i64*
586   %malloccall365 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
587   %factual366 = bitcast i8* %malloccall365 to double*
588   %ptwtointforfpstorage367 = ptrtoint double* %factual366 to i64
589   store double 1.000000e+00, double* %factual366
590   store i64 %ptwtointforfpstorage367, i64* %pointercast364
591   %ptrtointfornlstorage368 = ptrtoint i64* %fstorage337 to i64
592   store i64 %ptrtointfornlstorage368, i64* %nextptrptr334
593   %ptwtointfornlstorage369 = ptrtoint i64* %nextptrptr334 to i64
594   %nextptr370 = add i64 %ptwtointfornlstorage369, 8
595   %nextptrptr371 = inttoptr i64 %nextptr370 to i64*
596   %12 = trunc i64 5 to i32
597   %mallocsize372 = mul i32 %12, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
598   %malloccall373 = tail call i8* @malloc(i32 %mallocsize372)
599   %fstorage374 = bitcast i8* %malloccall373 to i64*
600   %intcast375 = ptrtoint i64* %fstorage374 to i64
601   %pmath376 = add i64 %intcast375, 0
602   %pointercast377 = inttoptr i64 %pmath376 to i64*
603   %malloccall378 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
604   %factual379 = bitcast i8* %malloccall378 to double*
605   %ptwtointforfpstorage380 = ptrtoint double* %factual379 to i64
606   store double 1.000000e+00, double* %factual379
607   store i64 %ptwtointforfpstorage380, i64* %pointercast377
608   %intcast381 = ptrtoint i64* %fstorage374 to i64
609   %pmath382 = add i64 %intcast381, 8
610   %pointercast383 = inttoptr i64 %pmath382 to i64*
611   %malloccall384 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
612   %factual385 = bitcast i8* %malloccall384 to double*
613   %ptwtointforfpstorage386 = ptrtoint double* %factual385 to i64
614   store double 0.000000e+00, double* %factual385
615   store i64 %ptwtointforfpstorage386, i64* %pointercast383
616   %intcast387 = ptrtoint i64* %fstorage374 to i64
617   %pmath388 = add i64 %intcast387, 16
618   %pointercast389 = inttoptr i64 %pmath388 to i64*
619   %malloccall390 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
620   %factual391 = bitcast i8* %malloccall390 to double*
621   %ptwtointforfpstorage392 = ptrtoint double* %factual391 to i64
622   store double 1.000000e+00, double* %factual391
623   store i64 %ptwtointforfpstorage392, i64* %pointercast389
```

```
  %intcast393 = ptrtoint i64* %fstorage374 to i64
  %pmath394 = add i64 %intcast393, 24
  %pointercast395 = inttoptr i64 %pmath394 to i64*
  %malloccall396 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual397 = bitcast i8* %malloccall396 to double*
  %ptwtointforfpstorage398 = ptrtoint double* %factual397 to i64
  store double 0.000000e+00, double* %factual397
  store i64 %ptwtointforfpstorage398, i64* %pointercast395
  %intcast399 = ptrtoint i64* %fstorage374 to i64
  %pmath400 = add i64 %intcast399, 32
  %pointercast401 = inttoptr i64 %pmath400 to i64*
  %malloccall402 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual403 = bitcast i8* %malloccall402 to double*
  %ptwtointforfpstorage404 = ptrtoint double* %factual403 to i64
  store double 1.000000e+00, double* %factual403
  store i64 %ptwtointforfpstorage404, i64* %pointercast401
  %ptrtointfornlstorage405 = ptrtoint i64* %fstorage374 to i64
  store i64 %ptrtointfornlstorage405, i64* %nextptrptr371
  %ptwtointfornlstorage406 = ptrtoint i64* %nextptrptr371 to i64
  %nextptr407 = add i64 %ptwtointfornlstorage406, 8
  %nextptrptr408 = inttoptr i64 %nextptr407 to i64*
  %13 = trunc i64 5 to i32
  %mallocsize409 = mul i32 %13, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall410 = tail call i8* @malloc(i32 %mallocsize409)
  %fstorage411 = bitcast i8* %malloccall410 to i64*
  %intcast412 = ptrtoint i64* %fstorage411 to i64
  %pmath413 = add i64 %intcast412, 0
  %pointercast414 = inttoptr i64 %pmath413 to i64*
  %malloccall415 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual416 = bitcast i8* %malloccall415 to double*
  %ptwtointforfpstorage417 = ptrtoint double* %factual416 to i64
  store double 1.000000e+00, double* %factual416
  store i64 %ptwtointforfpstorage417, i64* %pointercast414
  %intcast418 = ptrtoint i64* %fstorage411 to i64
  %pmath419 = add i64 %intcast418, 8
  %pointercast420 = inttoptr i64 %pmath419 to i64*
  %malloccall421 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual422 = bitcast i8* %malloccall421 to double*
  %ptwtointforfpstorage423 = ptrtoint double* %factual422 to i64
  store double 0.000000e+00, double* %factual422
  store i64 %ptwtointforfpstorage423, i64* %pointercast420
  %intcast424 = ptrtoint i64* %fstorage411 to i64
  %pmath425 = add i64 %intcast424, 16
  %pointercast426 = inttoptr i64 %pmath425 to i64*
  %malloccall427 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual428 = bitcast i8* %malloccall427 to double*
  %ptwtointforfpstorage429 = ptrtoint double* %factual428 to i64
  store double 1.000000e+00, double* %factual428
  store i64 %ptwtointforfpstorage429, i64* %pointercast426
  %intcast430 = ptrtoint i64* %fstorage411 to i64
  %pmath431 = add i64 %intcast430, 24
```

```
675   %pointercast432 = inttoptr i64 %pmath431 to i64*
676   %malloccall433 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
677   %factual434 = bitcast i8* %malloccall433 to double*
678   %ptwtointforfpstorage435 = ptrtoint double* %factual434 to i64
679   store double 1.000000e+00, double* %factual434
680   store i64 %ptwtointforfpstorage435, i64* %pointercast432
681   %intcast436 = ptrtoint i64* %fstorage411 to i64
682   %pmath437 = add i64 %intcast436, 32
683   %pointercast438 = inttoptr i64 %pmath437 to i64*
684   %malloccall439 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
685   %factual440 = bitcast i8* %malloccall439 to double*
686   %ptwtointforfpstorage441 = ptrtoint double* %factual440 to i64
687   store double 1.000000e+00, double* %factual440
688   store i64 %ptwtointforfpstorage441, i64* %pointercast438
689   %ptrtointfornlstorage442 = ptrtoint i64* %fstorage411 to i64
690   store i64 %ptrtointfornlstorage442, i64* %nextptrptr408
691   %ptwtointfornlstorage443 = ptrtoint i64* %nextptrptr408 to i64
692   %nextptr444 = add i64 %ptwtointfornlstorage443, 8
693   %nextptrptr445 = inttoptr i64 %nextptr444 to i64*
694   %14 = trunc i64 5 to i32
695   %mallocsize446 = mul i32 %14, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
696   %malloccall447 = tail call i8* @malloc(i32 %mallocsize446)
697   %fstorage448 = bitcast i8* %malloccall447 to i64*
698   %intcast449 = ptrtoint i64* %fstorage448 to i64
699   %pmath450 = add i64 %intcast449, 0
700   %pointercast451 = inttoptr i64 %pmath450 to i64*
701   %malloccall452 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
702   %factual453 = bitcast i8* %malloccall452 to double*
703   %ptwtointforfpstorage454 = ptrtoint double* %factual453 to i64
704   store double 1.000000e+00, double* %factual453
705   store i64 %ptwtointforfpstorage454, i64* %pointercast451
706   %intcast455 = ptrtoint i64* %fstorage448 to i64
707   %pmath456 = add i64 %intcast455, 8
708   %pointercast457 = inttoptr i64 %pmath456 to i64*
709   %malloccall458 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
710   %factual459 = bitcast i8* %malloccall458 to double*
711   %ptwtointforfpstorage460 = ptrtoint double* %factual459 to i64
712   store double 1.000000e+00, double* %factual459
713   store i64 %ptwtointforfpstorage460, i64* %pointercast457
714   %intcast461 = ptrtoint i64* %fstorage448 to i64
715   %pmath462 = add i64 %intcast461, 16
716   %pointercast463 = inttoptr i64 %pmath462 to i64*
717   %malloccall464 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
718   %factual465 = bitcast i8* %malloccall464 to double*
719   %ptwtointforfpstorage466 = ptrtoint double* %factual465 to i64
720   store double 0.000000e+00, double* %factual465
721   store i64 %ptwtointforfpstorage466, i64* %pointercast463
722   %intcast467 = ptrtoint i64* %fstorage448 to i64
723   %pmath468 = add i64 %intcast467, 24
724   %pointercast469 = inttoptr i64 %pmath468 to i64*
725   %malloccall470 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
```

```
        null, i32 1) to i32))
  %factual471 = bitcast i8* %malloccall470 to double*
  %ptwtointforfpstorage472 = ptrtoint double* %factual471 to i64
  store double 0.000000e+00, double* %factual471
  store i64 %ptwtointforfpstorage472, i64* %pointercast469
  %intcast473 = ptrtoint i64* %fstorage448 to i64
  %pmath474 = add i64 %intcast473, 32
  %pointercast475 = inttoptr i64 %pmath474 to i64*
  %malloccall476 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual477 = bitcast i8* %malloccall476 to double*
  %ptwtointforfpstorage478 = ptrtoint double* %factual477 to i64
  store double 1.000000e+00, double* %factual477
  store i64 %ptwtointforfpstorage478, i64* %pointercast475
  %ptrtointfornlstorage479 = ptrtoint i64* %fstorage448 to i64
  store i64 %ptrtointfornlstorage479, i64* %nextptrptr445
  %ptwtointfornlstorage480 = ptrtoint i64* %nextptrptr445 to i64
  %nextptr481 = add i64 %ptwtointfornlstorage480, 8
  %nextptrptr482 = inttoptr i64 %nextptr481 to i64*
  %15 = trunc i64 5 to i32
  %mallocsize483 = mul i32 %15, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall484 = tail call i8* @malloc(i32 %mallocsize483)
  %fstorage485 = bitcast i8* %malloccall484 to i64*
  %intcast486 = ptrtoint i64* %fstorage485 to i64
  %pmath487 = add i64 %intcast486, 0
  %pointercast488 = inttoptr i64 %pmath487 to i64*
  %malloccall489 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual490 = bitcast i8* %malloccall489 to double*
  %ptwtointforfpstorage491 = ptrtoint double* %factual490 to i64
  store double 1.000000e+00, double* %factual490
  store i64 %ptwtointforfpstorage491, i64* %pointercast488
  %intcast492 = ptrtoint i64* %fstorage485 to i64
  %pmath493 = add i64 %intcast492, 8
  %pointercast494 = inttoptr i64 %pmath493 to i64*
  %malloccall495 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual496 = bitcast i8* %malloccall495 to double*
  %ptwtointforfpstorage497 = ptrtoint double* %factual496 to i64
  store double 1.000000e+00, double* %factual496
  store i64 %ptwtointforfpstorage497, i64* %pointercast494
  %intcast498 = ptrtoint i64* %fstorage485 to i64
  %pmath499 = add i64 %intcast498, 16
  %pointercast500 = inttoptr i64 %pmath499 to i64*
  %malloccall501 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual502 = bitcast i8* %malloccall501 to double*
  %ptwtointforfpstorage503 = ptrtoint double* %factual502 to i64
  store double 0.000000e+00, double* %factual502
  store i64 %ptwtointforfpstorage503, i64* %pointercast500
  %intcast504 = ptrtoint i64* %fstorage485 to i64
  %pmath505 = add i64 %intcast504, 24
  %pointercast506 = inttoptr i64 %pmath505 to i64*
  %malloccall507 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual508 = bitcast i8* %malloccall507 to double*
```

```
776   %ptwtointforfpstorage509 = ptrtoint double* %factual508 to i64
777   store double 1.000000e+00, double* %factual508
778   store i64 %ptwtointforfpstorage509, i64* %pointercast506
779   %intcast510 = ptrtoint i64* %fstorage485 to i64
780   %pmath511 = add i64 %intcast510, 32
781   %pointercast512 = inttoptr i64 %pmath511 to i64*
782   %malloccall513 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
783   %factual514 = bitcast i8* %malloccall513 to double*
784   %ptwtointforfpstorage515 = ptrtoint double* %factual514 to i64
785   store double 1.000000e+00, double* %factual514
786   store i64 %ptwtointforfpstorage515, i64* %pointercast512
787   %ptrtointfornlstorage516 = ptrtoint i64* %fstorage485 to i64
788   store i64 %ptrtointfornlstorage516, i64* %nextptrptr482
789   %ptwtointfornlstorage517 = ptrtoint i64* %nextptrptr482 to i64
790   %nextptr518 = add i64 %ptwtointfornlstorage517, 8
791   %nextptrptr519 = inttoptr i64 %nextptr518 to i64*
792   %16 = trunc i64 5 to i32
793   %mallocsize520 = mul i32 %16, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
794   %malloccall521 = tail call i8* @malloc(i32 %mallocsize520)
795   %fstorage522 = bitcast i8* %malloccall521 to i64*
796   %intcast523 = ptrtoint i64* %fstorage522 to i64
797   %pmath524 = add i64 %intcast523, 0
798   %pointercast525 = inttoptr i64 %pmath524 to i64*
799   %malloccall526 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
800   %factual527 = bitcast i8* %malloccall526 to double*
801   %ptwtointforfpstorage528 = ptrtoint double* %factual527 to i64
802   store double 1.000000e+00, double* %factual527
803   store i64 %ptwtointforfpstorage528, i64* %pointercast525
804   %intcast529 = ptrtoint i64* %fstorage522 to i64
805   %pmath530 = add i64 %intcast529, 8
806   %pointercast531 = inttoptr i64 %pmath530 to i64*
807   %malloccall532 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
808   %factual533 = bitcast i8* %malloccall532 to double*
809   %ptwtointforfpstorage534 = ptrtoint double* %factual533 to i64
810   store double 1.000000e+00, double* %factual533
811   store i64 %ptwtointforfpstorage534, i64* %pointercast531
812   %intcast535 = ptrtoint i64* %fstorage522 to i64
813   %pmath536 = add i64 %intcast535, 16
814   %pointercast537 = inttoptr i64 %pmath536 to i64*
815   %malloccall538 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
816   %factual539 = bitcast i8* %malloccall538 to double*
817   %ptwtointforfpstorage540 = ptrtoint double* %factual539 to i64
818   store double 1.000000e+00, double* %factual539
819   store i64 %ptwtointforfpstorage540, i64* %pointercast537
820   %intcast541 = ptrtoint i64* %fstorage522 to i64
821   %pmath542 = add i64 %intcast541, 24
822   %pointercast543 = inttoptr i64 %pmath542 to i64*
823   %malloccall544 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
824   %factual545 = bitcast i8* %malloccall544 to double*
825   %ptwtointforfpstorage546 = ptrtoint double* %factual545 to i64
826   store double 0.000000e+00, double* %factual545
```

```
827   store i64 %ptwtointforfpstorage546, i64* %pointercast543
828   %intcast547 = ptrtoint i64* %fstorage522 to i64
829   %pmath548 = add i64 %intcast547, 32
830   %pointercast549 = inttoptr i64 %pmath548 to i64*
831   %malloccall550 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
832   %factual551 = bitcast i8* %malloccall550 to double*
833   %ptwtointforfpstorage552 = ptrtoint double* %factual551 to i64
834   store double 1.000000e+00, double* %factual551
835   store i64 %ptwtointforfpstorage552, i64* %pointercast549
836   %ptrtointfornlstorage553 = ptrtoint i64* %fstorage522 to i64
837   store i64 %ptrtointfornlstorage553, i64* %nextptrptr519
838   %ptwtointfornlstorage554 = ptrtoint i64* %nextptrptr519 to i64
839   %nextptr555 = add i64 %ptwtointfornlstorage554, 8
840   %nextptrptr556 = inttoptr i64 %nextptr555 to i64*
841   %17 = trunc i64 5 to i32
842   %mallocsize557 = mul i32 %17, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
843   %malloccall558 = tail call i8* @malloc(i32 %mallocsize557)
844   %fstorage559 = bitcast i8* %malloccall558 to i64*
845   %intcast560 = ptrtoint i64* %fstorage559 to i64
846   %pmath561 = add i64 %intcast560, 0
847   %pointercast562 = inttoptr i64 %pmath561 to i64*
848   %malloccall563 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
849   %factual564 = bitcast i8* %malloccall563 to double*
850   %ptwtointforfpstorage565 = ptrtoint double* %factual564 to i64
851   store double 1.000000e+00, double* %factual564
852   store i64 %ptwtointforfpstorage565, i64* %pointercast562
853   %intcast566 = ptrtoint i64* %fstorage559 to i64
854   %pmath567 = add i64 %intcast566, 8
855   %pointercast568 = inttoptr i64 %pmath567 to i64*
856   %malloccall569 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
857   %factual570 = bitcast i8* %malloccall569 to double*
858   %ptwtointforfpstorage571 = ptrtoint double* %factual570 to i64
859   store double 1.000000e+00, double* %factual570
860   store i64 %ptwtointforfpstorage571, i64* %pointercast568
861   %intcast572 = ptrtoint i64* %fstorage559 to i64
862   %pmath573 = add i64 %intcast572, 16
863   %pointercast574 = inttoptr i64 %pmath573 to i64*
864   %malloccall575 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
865   %factual576 = bitcast i8* %malloccall575 to double*
866   %ptwtointforfpstorage577 = ptrtoint double* %factual576 to i64
867   store double 1.000000e+00, double* %factual576
868   store i64 %ptwtointforfpstorage577, i64* %pointercast574
869   %intcast578 = ptrtoint i64* %fstorage559 to i64
870   %pmath579 = add i64 %intcast578, 24
871   %pointercast580 = inttoptr i64 %pmath579 to i64*
872   %malloccall581 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
873   %factual582 = bitcast i8* %malloccall581 to double*
874   %ptwtointforfpstorage583 = ptrtoint double* %factual582 to i64
875   store double 1.000000e+00, double* %factual582
876   store i64 %ptwtointforfpstorage583, i64* %pointercast580
877   %intcast584 = ptrtoint i64* %fstorage559 to i64
```

```
878   %pmath585 = add i64 %intcast584, 32
879   %pointercast586 = inttoptr i64 %pmath585 to i64*
880   %malloccall587 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
881   %factual588 = bitcast i8* %malloccall587 to double*
882   %ptwtointforfpstorage589 = ptrtoint double* %factual588 to i64
883   store double 1.000000e+00, double* %factual588
884   store i64 %ptwtointforfpstorage589, i64* %pointercast586
885   %ptrtointfornlstorage590 = ptrtoint i64* %fstorage559 to i64
886   store i64 %ptrtointfornlstorage590, i64* %nextptrptr556
887   %ptwtointfornlstorage591 = ptrtoint i64* %nextptrptr556 to i64
888   %nextptr592 = add i64 %ptwtointfornlstorage591, 8
889   %nextptrptr593 = inttoptr i64 %nextptr592 to i64*
890   %ptrtointfornlstarstorage = ptrtoint i64* %fstarstorage to i64
891   store i64 %ptrtointfornlstarstorage, i64* %pointercast6
892   %ptwtointfornlstorage594 = ptrtoint i64* %pointercast6 to i64
893   %nextptr595 = add i64 %ptwtointfornlstorage594, 8
894   %nextptrptr596 = inttoptr i64 %nextptr595 to i64*
895   store i64* %immediate_tmp, i64** %trainingData
896   %trainingLabels = alloca i64*
897   %18 = trunc i64 3 to i32
898   %mallocsize597 = mul i32 %18, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
899   %malloccall598 = tail call i8* @malloc(i32 %mallocsize597)
900   %immediate_tmp599 = bitcast i8* %malloccall598 to i64*
901   store i64 1, i64* %immediate_tmp599
902   %intcast600 = ptrtoint i64* %immediate_tmp599 to i64
903   %pmath601 = add i64 %intcast600, 8
904   %pointercast602 = inttoptr i64 %pmath601 to i64*
905   store i64 16, i64* %pointercast602
906   %intcast603 = ptrtoint i64* %immediate_tmp599 to i64
907   %pmath604 = add i64 %intcast603, 16
908   %pointercast605 = inttoptr i64 %pmath604 to i64*
909   %19 = trunc i64 16 to i32
910   %mallocsize606 = mul i32 %19, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
911   %malloccall607 = tail call i8* @malloc(i32 %mallocsize606)
912   %fstorage608 = bitcast i8* %malloccall607 to i64*
913   %intcast609 = ptrtoint i64* %fstorage608 to i64
914   %pmath610 = add i64 %intcast609, 0
915   %pointercast611 = inttoptr i64 %pmath610 to i64*
916   %malloccall612 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
917   %factual613 = bitcast i8* %malloccall612 to double*
918   %ptwtointforfpstorage614 = ptrtoint double* %factual613 to i64
919   store double -1.000000e+00, double* %factual613
920   store i64 %ptwtointforfpstorage614, i64* %pointercast611
921   %intcast615 = ptrtoint i64* %fstorage608 to i64
922   %pmath616 = add i64 %intcast615, 8
923   %pointercast617 = inttoptr i64 %pmath616 to i64*
924   %malloccall618 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
925   %factual619 = bitcast i8* %malloccall618 to double*
926   %ptwtointforfpstorage620 = ptrtoint double* %factual619 to i64
927   store double -1.000000e+00, double* %factual619
928   store i64 %ptwtointforfpstorage620, i64* %pointercast617
929   %intcast621 = ptrtoint i64* %fstorage608 to i64
930   %pmath622 = add i64 %intcast621, 16
```

```
931   %pointercast623 = inttoptr i64 %pmath622 to i64*
932   %malloccall624 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
933   %factual625 = bitcast i8* %malloccall624 to double*
934   %ptwtointforfpstorage626 = ptrtoint double* %factual625 to i64
935   store double -1.000000e+00, double* %factual625
936   store i64 %ptwtointforfpstorage626, i64* %pointercast623
937   %intcast627 = ptrtoint i64* %fstorage608 to i64
938   %pmath628 = add i64 %intcast627, 24
939   %pointercast629 = inttoptr i64 %pmath628 to i64*
940   %malloccall630 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
941   %factual631 = bitcast i8* %malloccall630 to double*
942   %ptwtointforfpstorage632 = ptrtoint double* %factual631 to i64
943   store double 1.000000e+00, double* %factual631
944   store i64 %ptwtointforfpstorage632, i64* %pointercast629
945   %intcast633 = ptrtoint i64* %fstorage608 to i64
946   %pmath634 = add i64 %intcast633, 32
947   %pointercast635 = inttoptr i64 %pmath634 to i64*
948   %malloccall636 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
949   %factual637 = bitcast i8* %malloccall636 to double*
950   %ptwtointforfpstorage638 = ptrtoint double* %factual637 to i64
951   store double -1.000000e+00, double* %factual637
952   store i64 %ptwtointforfpstorage638, i64* %pointercast635
953   %intcast639 = ptrtoint i64* %fstorage608 to i64
954   %pmath640 = add i64 %intcast639, 40
955   %pointercast641 = inttoptr i64 %pmath640 to i64*
956   %malloccall642 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
957   %factual643 = bitcast i8* %malloccall642 to double*
958   %ptwtointforfpstorage644 = ptrtoint double* %factual643 to i64
959   store double -1.000000e+00, double* %factual643
960   store i64 %ptwtointforfpstorage644, i64* %pointercast641
961   %intcast645 = ptrtoint i64* %fstorage608 to i64
962   %pmath646 = add i64 %intcast645, 48
963   %pointercast647 = inttoptr i64 %pmath646 to i64*
964   %malloccall648 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
965   %factual649 = bitcast i8* %malloccall648 to double*
966   %ptwtointforfpstorage650 = ptrtoint double* %factual649 to i64
967   store double -1.000000e+00, double* %factual649
968   store i64 %ptwtointforfpstorage650, i64* %pointercast647
969   %intcast651 = ptrtoint i64* %fstorage608 to i64
970   %pmath652 = add i64 %intcast651, 56
971   %pointercast653 = inttoptr i64 %pmath652 to i64*
972   %malloccall654 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
          null, i32 1) to i32))
973   %factual655 = bitcast i8* %malloccall654 to double*
974   %ptwtointforfpstorage656 = ptrtoint double* %factual655 to i64
975   store double 1.000000e+00, double* %factual655
976   store i64 %ptwtointforfpstorage656, i64* %pointercast653
977   %intcast657 = ptrtoint i64* %fstorage608 to i64
978   %pmath658 = add i64 %intcast657, 64
979   %pointercast659 = inttoptr i64 %pmath658 to i64*
980   %malloccall660 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
```

```
        null, i32 1) to i32))
  %factual661 = bitcast i8* %malloccall660 to double*
  %ptwtointforfpstorage662 = ptrtoint double* %factual661 to i64
  store double -1.000000e+00, double* %factual661
  store i64 %ptwtointforfpstorage662, i64* %pointercast659
  %intcast663 = ptrtoint i64* %fstorage608 to i64
  %pmath664 = add i64 %intcast663, 72
  %pointercast665 = inttoptr i64 %pmath664 to i64*
  %malloccall666 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual667 = bitcast i8* %malloccall666 to double*
  %ptwtointforfpstorage668 = ptrtoint double* %factual667 to i64
  store double -1.000000e+00, double* %factual667
  store i64 %ptwtointforfpstorage668, i64* %pointercast665
  %intcast669 = ptrtoint i64* %fstorage608 to i64
  %pmath670 = add i64 %intcast669, 80
  %pointercast671 = inttoptr i64 %pmath670 to i64*
  %malloccall672 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual673 = bitcast i8* %malloccall672 to double*
  %ptwtointforfpstorage674 = ptrtoint double* %factual673 to i64
  store double -1.000000e+00, double* %factual673
  store i64 %ptwtointforfpstorage674, i64* %pointercast671
  %intcast675 = ptrtoint i64* %fstorage608 to i64
  %pmath676 = add i64 %intcast675, 88
  %pointercast677 = inttoptr i64 %pmath676 to i64*
  %malloccall678 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual679 = bitcast i8* %malloccall678 to double*
  %ptwtointforfpstorage680 = ptrtoint double* %factual679 to i64
  store double 1.000000e+00, double* %factual679
  store i64 %ptwtointforfpstorage680, i64* %pointercast677
  %intcast681 = ptrtoint i64* %fstorage608 to i64
  %pmath682 = add i64 %intcast681, 96
  %pointercast683 = inttoptr i64 %pmath682 to i64*
  %malloccall684 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual685 = bitcast i8* %malloccall684 to double*
  %ptwtointforfpstorage686 = ptrtoint double* %factual685 to i64
  store double 1.000000e+00, double* %factual685
  store i64 %ptwtointforfpstorage686, i64* %pointercast683
  %intcast687 = ptrtoint i64* %fstorage608 to i64
  %pmath688 = add i64 %intcast687, 104
  %pointercast689 = inttoptr i64 %pmath688 to i64*
  %malloccall690 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual691 = bitcast i8* %malloccall690 to double*
  %ptwtointforfpstorage692 = ptrtoint double* %factual691 to i64
  store double 1.000000e+00, double* %factual691
  store i64 %ptwtointforfpstorage692, i64* %pointercast689
  %intcast693 = ptrtoint i64* %fstorage608 to i64
  %pmath694 = add i64 %intcast693, 112
  %pointercast695 = inttoptr i64 %pmath694 to i64*
  %malloccall696 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
        null, i32 1) to i32))
  %factual697 = bitcast i8* %malloccall696 to double*
```

```
  %ptwtointforfpstorage698 = ptrtoint double* %factual697 to i64
  store double 1.000000e+00, double* %factual697
  store i64 %ptwtointforfpstorage698, i64* %pointercast695
  %intcast699 = ptrtoint i64* %fstorage608 to i64
  %pmath700 = add i64 %intcast699, 120
  %pointercast701 = inttoptr i64 %pmath700 to i64*
  %malloccall702 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double*
      null, i32 1) to i32))
  %factual703 = bitcast i8* %malloccall702 to double*
  %ptwtointforfpstorage704 = ptrtoint double* %factual703 to i64
  store double 1.000000e+00, double* %factual703
  store i64 %ptwtointforfpstorage704, i64* %pointercast701
  %ptrtointfornlstorage705 = ptrtoint i64* %fstorage608 to i64
  store i64 %ptrtointfornlstorage705, i64* %pointercast605
  %ptwtointfornlstorage706 = ptrtoint i64* %pointercast605 to i64
  %nextptr707 = add i64 %ptwtointfornlstorage706, 8
  %nextptrptr708 = inttoptr i64 %nextptr707 to i64*
  store i64* %immediate_tmp599, i64** %trainingLabels
  %20 = trunc i64 3 to i32
  %mallocsize709 = mul i32 %20, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall710 = tail call i8* @malloc(i32 %mallocsize709)
  %dynamic_ten = bitcast i8* %malloccall710 to i64*
  store i64 1, i64* %dynamic_ten
  %trainingData711 = load i64*, i64** %trainingData
  %intcast712 = ptrtoint i64* %trainingData711 to i64
  %pmath713 = add i64 %intcast712, 16
  %pointercast714 = inttoptr i64 %pmath713 to i64*
  %tenresult = load i64, i64* %pointercast714
  %intcast715 = ptrtoint i64* %dynamic_ten to i64
  %pmath716 = add i64 %intcast715, 8
  %pointercast717 = inttoptr i64 %pmath716 to i64*
  store i64 %tenresult, i64* %pointercast717
  %_free_vars1 = alloca i64
  %intcast718 = ptrtoint i64* %dynamic_ten to i64
  %pmath719 = add i64 %intcast718, 8
  %pointercast720 = inttoptr i64 %pmath719 to i64*
  %slicesizeload = load i64, i64* %pointercast720
  %offsetload = load i64, i64* %dynamic_ten
  %plus1 = add i64 %offsetload, 1
  %intcast721 = ptrtoint i64* %dynamic_ten to i64
  %offsetcalc = mul i64 %plus1, 8
  %pmath722 = add i64 %intcast721, %offsetcalc
  %pointercast723 = inttoptr i64 %pmath722 to i64*
  %21 = trunc i64 %slicesizeload to i32
  %mallocsize724 = mul i32 %21, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall725 = tail call i8* @malloc(i32 %mallocsize724)
  %mcarr = bitcast i8* %malloccall725 to i64*
  %mcarrptrtoint = ptrtoint i64* %mcarr to i64
  store i64 %mcarrptrtoint, i64* %pointercast723
  store i64 0, i64* %_free_vars1
  br label %while

while:                                            ; preds = %while_body, %entry
  %_free_vars1737 = load i64, i64* %_free_vars1
  %_slicesize_0 = load i64, i64* %pointercast720
  %ftmp = sitofp i64 %_slicesize_0 to double
```

```
  %ftmp738 = sitofp i64 %_free_vars1737 to double
  %tmp739 = fcmp olt double %ftmp738, %ftmp
  %booltmp = icmp ne i1 %tmp739, false
  br i1 %booltmp, label %while_body, label %merge

while_body:                                       ; preds = %while
  %gl = load i64, i64* %_free_vars1
  %offsetload726 = load i64, i64* %dynamic_ten
  %plus1727 = add i64 %offsetload726, 1
  %intcast728 = ptrtoint i64* %dynamic_ten to i64
  %offsetcalc729 = mul i64 %plus1727, 8
  %pmath730 = add i64 %intcast728, %offsetcalc729
  %pointercast731 = inttoptr i64 %pmath730 to i64*
  %deref = load i64, i64* %pointercast731
  %offsetcalc732 = mul i64 %gl, 8
  %pmath733 = add i64 %deref, %offsetcalc732
  %pointercast734 = inttoptr i64 %pmath733 to i64*
  %malloccall735 = tail call i8* @malloc(i32 ptrtoint (i64* getelementptr (i64, i64* null, i32 1)
      to i32))
  %abldrk = bitcast i8* %malloccall735 to i64*
  %abldrkcast = ptrtoint i64* %abldrk to i64
  store i64 %abldrkcast, i64* %pointercast734
  %_free_vars1736 = load i64, i64* %_free_vars1
  %tmp = add i64 1, %_free_vars1736
  store i64 %tmp, i64* %_free_vars1
  br label %while

merge:                                            ; preds = %while
  %theta = alloca i64*
  store i64* %dynamic_ten, i64** %theta
  %check = alloca i64
  store i64 1, i64* %check
  %count = alloca i64
  store i64 0, i64* %count
  %mistakes = alloca i64
  store i64 0, i64* %mistakes
  br label %while740

while740:                                         ; preds = %merge976, %merge
  %check977 = load i64, i64* %check
  %tmp978 = icmp eq i64 %check977, 1
  %count979 = load i64, i64* %count
  %ftmp980 = sitofp i64 %count979 to double
  %tmp981 = fcmp olt double %ftmp980, 5.000000e+02
  %booltmp982 = icmp ne i1 %tmp978, false
  %booltmp983 = icmp ne i1 %tmp981, false
  %tmp984 = and i1 %booltmp982, %booltmp983
  %booltmp985 = icmp ne i1 %tmp984, false
  br i1 %booltmp985, label %while_body741, label %merge986

while_body741:                                    ; preds = %while740
  store i64 0, i64* %check
  %count742 = load i64, i64* %count
  %tmp743 = add i64 %count742, 1
  store i64 %tmp743, i64* %count
  %i = alloca i64
```

```
  store i64 0, i64* %i
  br label %while744

while744:                                         ; preds = %merge890, %while_body741
  %i966 = load i64, i64* %i
  %trainingLabels967 = load i64*, i64** %trainingLabels
  %intcast968 = ptrtoint i64* %trainingLabels967 to i64
  %pmath969 = add i64 %intcast968, 8
  %pointercast970 = inttoptr i64 %pmath969 to i64*
  %tenresult971 = load i64, i64* %pointercast970
  %ftmp972 = sitofp i64 %tenresult971 to double
  %ftmp973 = sitofp i64 %i966 to double
  %tmp974 = fcmp olt double %ftmp973, %ftmp972
  %booltmp975 = icmp ne i1 %tmp974, false
  br i1 %booltmp975, label %while_body745, label %merge976

while_body745:                                    ; preds = %while744
  %y = alloca double
  %i746 = load i64, i64* %i
  %free_alloc = alloca i64
  store i64 %i746, i64* %free_alloc
  %trainingLabels747 = load i64*, i64** %trainingLabels
  %glr = load i64, i64* %free_alloc
  %offsetload748 = load i64, i64* %trainingLabels747
  %plus1749 = add i64 %offsetload748, 1
  %intcast750 = ptrtoint i64* %trainingLabels747 to i64
  %offsetcalc751 = mul i64 %plus1749, 8
  %pmath752 = add i64 %intcast750, %offsetcalc751
  %pointercast753 = inttoptr i64 %pmath752 to i64*
  %rootload = load i64, i64* %pointercast753
  %offsetcalc754 = mul i64 %glr, 8
  %pmath755 = add i64 %rootload, %offsetcalc754
  %pointercast756 = inttoptr i64 %pmath755 to i64*
  %deref757 = load i64, i64* %pointercast756
  %derefpntr = inttoptr i64 %deref757 to i64*
  %fpaccess = bitcast i64* %derefpntr to double*
  %fpret = load double, double* %fpaccess
  store double %fpret, double* %y
  %x = alloca i64*
  %i758 = load i64, i64* %i
  %free_alloc759 = alloca i64
  store i64 %i758, i64* %free_alloc759
  %free_alloc760 = alloca i64
  store i64 0, i64* %free_alloc760
  %trainingData761 = load i64*, i64** %trainingData
  %22 = trunc i64 3 to i32
  %mallocsize762 = mul i32 %22, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall763 = tail call i8* @malloc(i32 %mallocsize762)
  %cpy_hdr = bitcast i8* %malloccall763 to i64*
  store i64 1, i64* %cpy_hdr
  %tgt_header = alloca i64*
  store i64* %cpy_hdr, i64** %tgt_header
  %srcint = ptrtoint i64* %trainingData761 to i64
  %dstint = ptrtoint i64* %cpy_hdr to i64
  %sintp1 = add i64 %srcint, 8
  %dintp1 = add i64 %dstint, 8
```

```
  %srcint764 = add i64 %sintp1, 8
  %srcptr = inttoptr i64 %srcint764 to i64*
  %dstptr = inttoptr i64 %dintp1 to i64*
  %memload = load i64, i64* %srcptr
  store i64 %memload, i64* %dstptr
  %srcint765 = add i64 %srcint764, 8
  %dstint766 = add i64 %dintp1, 8
  %intcast767 = ptrtoint i64* %trainingData761 to i64
  %pmath768 = add i64 %intcast767, 8
  %pointercast769 = inttoptr i64 %pmath768 to i64*
  %intcast770 = ptrtoint i64* %trainingData761 to i64
  %pmath771 = add i64 %intcast770, 16
  %pointercast772 = inttoptr i64 %pmath771 to i64*
  %slicesizeload773 = load i64, i64* %pointercast772
  %offsetload774 = load i64, i64* %cpy_hdr
  %plus1775 = add i64 %offsetload774, 1
  %intcast776 = ptrtoint i64* %cpy_hdr to i64
  %offsetcalc777 = mul i64 %plus1775, 8
  %pmath778 = add i64 %intcast776, %offsetcalc777
  %pointercast779 = inttoptr i64 %pmath778 to i64*
  %23 = trunc i64 %slicesizeload773 to i32
  %mallocsize780 = mul i32 %23, ptrtoint (i64* getelementptr (i64, i64* null, i32 1) to i32)
  %malloccall781 = tail call i8* @malloc(i32 %mallocsize780)
  %mcarr782 = bitcast i8* %malloccall781 to i64*
  %mcarrptrtoint783 = ptrtoint i64* %mcarr782 to i64
  store i64 %mcarrptrtoint783, i64* %pointercast779
  store i64 0, i64* %free_alloc760
  br label %while784

while784:                                         ; preds = %while_body785, %while_body745
  %_slicenot_1824 = load i64, i64* %free_alloc760
  %_slicesize_0825 = load i64, i64* %pointercast772
  %ftmp826 = sitofp i64 %_slicesize_0825 to double
  %ftmp827 = sitofp i64 %_slicenot_1824 to double
  %tmp828 = fcmp olt double %ftmp827, %ftmp826
  %booltmp829 = icmp ne i1 %tmp828, false
  br i1 %booltmp829, label %while_body785, label %merge830

while_body785:                                    ; preds = %while784
  %_slicenot_0 = load i64, i64* %free_alloc759
  %free_alloc786 = alloca i64
  store i64 %_slicenot_0, i64* %free_alloc786
  %_slicenot_1 = load i64, i64* %free_alloc760
  %free_alloc787 = alloca i64
  store i64 %_slicenot_1, i64* %free_alloc787
  %trainingData788 = load i64*, i64** %trainingData
  %glr789 = load i64, i64* %free_alloc787
  %glr790 = load i64, i64* %free_alloc786
  %offsetload791 = load i64, i64* %trainingData788
  %plus1792 = add i64 %offsetload791, 1
  %intcast793 = ptrtoint i64* %trainingData788 to i64
  %offsetcalc794 = mul i64 %plus1792, 8
  %pmath795 = add i64 %intcast793, %offsetcalc794
  %pointercast796 = inttoptr i64 %pmath795 to i64*
  %rootload797 = load i64, i64* %pointercast796
  %offsetcalc798 = mul i64 %glr790, 8
```

```
  %pmath799 = add i64 %rootload797, %offsetcalc798
  %pointercast800 = inttoptr i64 %pmath799 to i64*
  %deref801 = load i64, i64* %pointercast800
  %derefpntr802 = inttoptr i64 %deref801 to i64*
  %intcast803 = ptrtoint i64* %derefpntr802 to i64
  %offsetcalc804 = mul i64 %glr789, 8
  %pmath805 = add i64 %intcast803, %offsetcalc804
  %pointercast806 = inttoptr i64 %pmath805 to i64*
  %deref807 = load i64, i64* %pointercast806
  %derefpntr808 = inttoptr i64 %deref807 to i64*
  %icast = ptrtoint i64* %derefpntr808 to i64
  %_slicenot_1809 = load i64, i64* %free_alloc760
  %free_alloc810 = alloca i64
  store i64 %_slicenot_1809, i64* %free_alloc810
  %_tgt_header = load i64*, i64** %tgt_header
  %glr811 = load i64, i64* %free_alloc810
  %offsetload812 = load i64, i64* %_tgt_header
  %plus1813 = add i64 %offsetload812, 1
  %intcast814 = ptrtoint i64* %_tgt_header to i64
  %offsetcalc815 = mul i64 %plus1813, 8
  %pmath816 = add i64 %intcast814, %offsetcalc815
  %pointercast817 = inttoptr i64 %pmath816 to i64*
  %deref818 = load i64, i64* %pointercast817
  %offsetcalc819 = mul i64 %glr811, 8
  %pmath820 = add i64 %deref818, %offsetcalc819
  %pointercast821 = inttoptr i64 %pmath820 to i64*
  store i64 %icast, i64* %pointercast821
  %_slicenot_1822 = load i64, i64* %free_alloc760
  %tmp823 = add i64 1, %_slicenot_1822
  store i64 %tmp823, i64* %free_alloc760
  br label %while784

merge830:                                         ; preds = %while784
  store i64* %cpy_hdr, i64** %x
  %guess = alloca double
  %free_alloc831 = alloca i64
  store i64 0, i64* %free_alloc831
  %x832 = load i64*, i64** %x
  %free_alloc833 = alloca i64
  store i64 0, i64* %free_alloc833
  %theta834 = load i64*, i64** %theta
  %intcast835 = ptrtoint i64* %x832 to i64
  %pmath836 = add i64 %intcast835, 8
  %pointercast837 = inttoptr i64 %pmath836 to i64*
  %tgt_header838 = alloca double
  %s1_header = alloca i64*
  %s2_header = alloca i64*
  store double 0.000000e+00, double* %tgt_header838
  store i64* %x832, i64** %s1_header
  store i64* %theta834, i64** %s2_header
  %a = alloca i64
  store double 0.000000e+00, double* %tgt_header838
  store i64 0, i64* %a
  br label %while839

while839:                                         ; preds = %while_body840, %merge830
```

```
  %a879 = load i64, i64* %a
  %_tmult_size_inner_0 = load i64, i64* %pointercast837
  %ftmp880 = sitofp i64 %_tmult_size_inner_0 to double
  %ftmp881 = sitofp i64 %a879 to double
  %tmp882 = fcmp olt double %ftmp881, %ftmp880
  %booltmp883 = icmp ne i1 %tmp882, false
  br i1 %booltmp883, label %while_body840, label %merge884

while_body840:                                    ; preds = %while839
  %_tgt_header_ = load double, double* %tgt_header838
  %a841 = load i64, i64* %a
  %free_alloc842 = alloca i64
  store i64 %a841, i64* %free_alloc842
  %_s1_header_ = load i64*, i64** %s1_header
  %glr843 = load i64, i64* %free_alloc842
  %offsetload844 = load i64, i64* %_s1_header_
  %plus1845 = add i64 %offsetload844, 1
  %intcast846 = ptrtoint i64* %_s1_header_ to i64
  %offsetcalc847 = mul i64 %plus1845, 8
  %pmath848 = add i64 %intcast846, %offsetcalc847
  %pointercast849 = inttoptr i64 %pmath848 to i64*
  %rootload850 = load i64, i64* %pointercast849
  %offsetcalc851 = mul i64 %glr843, 8
  %pmath852 = add i64 %rootload850, %offsetcalc851
  %pointercast853 = inttoptr i64 %pmath852 to i64*
  %deref854 = load i64, i64* %pointercast853
  %derefpntr855 = inttoptr i64 %deref854 to i64*
  %fpaccess856 = bitcast i64* %derefpntr855 to double*
  %fpret857 = load double, double* %fpaccess856
  %a858 = load i64, i64* %a
  %free_alloc859 = alloca i64
  store i64 %a858, i64* %free_alloc859
  %_s2_header_ = load i64*, i64** %s2_header
  %glr860 = load i64, i64* %free_alloc859
  %offsetload861 = load i64, i64* %_s2_header_
  %plus1862 = add i64 %offsetload861, 1
  %intcast863 = ptrtoint i64* %_s2_header_ to i64
  %offsetcalc864 = mul i64 %plus1862, 8
  %pmath865 = add i64 %intcast863, %offsetcalc864
  %pointercast866 = inttoptr i64 %pmath865 to i64*
  %rootload867 = load i64, i64* %pointercast866
  %offsetcalc868 = mul i64 %glr860, 8
  %pmath869 = add i64 %rootload867, %offsetcalc868
  %pointercast870 = inttoptr i64 %pmath869 to i64*
  %deref871 = load i64, i64* %pointercast870
  %derefpntr872 = inttoptr i64 %deref871 to i64*
  %fpaccess873 = bitcast i64* %derefpntr872 to double*
  %fpret874 = load double, double* %fpaccess873
  %tmp875 = fmul double %fpret857, %fpret874
  %tmp876 = fadd double %_tgt_header_, %tmp875
  store double %tmp876, double* %tgt_header838
  %a877 = load i64, i64* %a
  %tmp878 = add i64 1, %a877
  store i64 %tmp878, i64* %a
  br label %while839

```

```
merge884:                                         ; preds = %while839
  %hdraddr_load = load double, double* %tgt_header838
  store double %hdraddr_load, double* %guess
  %guess885 = load double, double* %guess
  %y886 = load double, double* %y
  %tmp887 = fmul double %guess885, %y886
  %tmp888 = fcmp ole double %tmp887, 0.000000e+00
  %booltmp889 = icmp ne i1 %tmp888, false
  br i1 %booltmp889, label %then, label %else

merge890:                                         ; preds = %else, %merge963
  %i964 = load i64, i64* %i
  %tmp965 = add i64 %i964, 1
  store i64 %tmp965, i64* %i
  br label %while744

then:                                             ; preds = %merge884
  store i64 1, i64* %check
  %mistakes891 = load i64, i64* %mistakes
  %tmp892 = add i64 %mistakes891, 1
  store i64 %tmp892, i64* %mistakes
  %j = alloca i64
  store i64 0, i64* %j
  br label %while893

while893:                                         ; preds = %while_body894, %then
  %j953 = load i64, i64* %j
  %theta954 = load i64*, i64** %theta
  %intcast955 = ptrtoint i64* %theta954 to i64
  %pmath956 = add i64 %intcast955, 8
  %pointercast957 = inttoptr i64 %pmath956 to i64*
  %tenresult958 = load i64, i64* %pointercast957
  %ftmp959 = sitofp i64 %tenresult958 to double
  %ftmp960 = sitofp i64 %j953 to double
  %tmp961 = fcmp olt double %ftmp960, %ftmp959
  %booltmp962 = icmp ne i1 %tmp961, false
  br i1 %booltmp962, label %while_body894, label %merge963

while_body894:                                    ; preds = %while893
  %j895 = load i64, i64* %j
  %free_alloc896 = alloca i64
  store i64 %j895, i64* %free_alloc896
  %theta897 = load i64*, i64** %theta
  %glr898 = load i64, i64* %free_alloc896
  %offsetload899 = load i64, i64* %theta897
  %plus1900 = add i64 %offsetload899, 1
  %intcast901 = ptrtoint i64* %theta897 to i64
  %offsetcalc902 = mul i64 %plus1900, 8
  %pmath903 = add i64 %intcast901, %offsetcalc902
  %pointercast904 = inttoptr i64 %pmath903 to i64*
  %rootload905 = load i64, i64* %pointercast904
  %offsetcalc906 = mul i64 %glr898, 8
  %pmath907 = add i64 %rootload905, %offsetcalc906
  %pointercast908 = inttoptr i64 %pmath907 to i64*
  %deref909 = load i64, i64* %pointercast908
  %derefpntr910 = inttoptr i64 %deref909 to i64*
```

```
  %fpaccess911 = bitcast i64* %derefpntr910 to double*
  %fpret912 = load double, double* %fpaccess911
  %y913 = load double, double* %y
  %j914 = load i64, i64* %j
  %free_alloc915 = alloca i64
  store i64 %j914, i64* %free_alloc915
  %x916 = load i64*, i64** %x
  %glr917 = load i64, i64* %free_alloc915
  %offsetload918 = load i64, i64* %x916
  %plus1919 = add i64 %offsetload918, 1
  %intcast920 = ptrtoint i64* %x916 to i64
  %offsetcalc921 = mul i64 %plus1919, 8
  %pmath922 = add i64 %intcast920, %offsetcalc921
  %pointercast923 = inttoptr i64 %pmath922 to i64*
  %rootload924 = load i64, i64* %pointercast923
  %offsetcalc925 = mul i64 %glr917, 8
  %pmath926 = add i64 %rootload924, %offsetcalc925
  %pointercast927 = inttoptr i64 %pmath926 to i64*
  %deref928 = load i64, i64* %pointercast927
  %derefpntr929 = inttoptr i64 %deref928 to i64*
  %fpaccess930 = bitcast i64* %derefpntr929 to double*
  %fpret931 = load double, double* %fpaccess930
  %tmp932 = fmul double %y913, %fpret931
  %tmp933 = fadd double %fpret912, %tmp932
  %j934 = load i64, i64* %j
  %free_alloc935 = alloca i64
  store i64 %j934, i64* %free_alloc935
  %theta936 = load i64*, i64** %theta
  %glr937 = load i64, i64* %free_alloc935
  %offsetload938 = load i64, i64* %theta936
  %plus1939 = add i64 %offsetload938, 1
  %intcast940 = ptrtoint i64* %theta936 to i64
  %offsetcalc941 = mul i64 %plus1939, 8
  %pmath942 = add i64 %intcast940, %offsetcalc941
  %pointercast943 = inttoptr i64 %pmath942 to i64*
  %rootload944 = load i64, i64* %pointercast943
  %offsetcalc945 = mul i64 %glr937, 8
  %pmath946 = add i64 %rootload944, %offsetcalc945
  %pointercast947 = inttoptr i64 %pmath946 to i64*
  %deref948 = load i64, i64* %pointercast947
  %derefpntr949 = inttoptr i64 %deref948 to i64*
  %fpaccess950 = bitcast i64* %derefpntr949 to double*
  store double %tmp933, double* %fpaccess950
  %j951 = load i64, i64* %j
  %tmp952 = add i64 %j951, 1
  store i64 %tmp952, i64* %j
  br label %while893

merge963:                                         ; preds = %while893
  br label %merge890

else:                                             ; preds = %merge884
  br label %merge890

merge976:                                         ; preds = %while744
  br label %while740
```

```
1476
1477 merge986:                                         ; preds = %while740
1478   %printf_ignore = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([19 x i8], [19 x i8]*
           @"FINAL CLASSIFIER:\0A", i32 0, i32 0))
1479   %theta987 = load i64*, i64** %theta
1480   %_printf_prefix_1 = alloca i64
1481   %intcast988 = ptrtoint i64* %theta987 to i64
1482   %pmath989 = add i64 %intcast988, 8
1483   %pointercast990 = inttoptr i64 %pmath989 to i64*
1484   store i64 0, i64* %_printf_prefix_1
1485   br label %while991
1486
1487 while991:                                         ; preds = %while_body992, %merge986
1488   %_printf_prefix_11011 = load i64, i64* %_printf_prefix_1
1489   %_slicesize_01012 = load i64, i64* %pointercast990
1490   %ftmp1013 = sitofp i64 %_slicesize_01012 to double
1491   %ftmp1014 = sitofp i64 %_printf_prefix_11011 to double
1492   %tmp1015 = fcmp olt double %ftmp1014, %ftmp1013
1493   %booltmp1016 = icmp ne i1 %tmp1015, false
1494   br i1 %booltmp1016, label %while_body992, label %merge1017
1495
1496 while_body992:                                    ; preds = %while991
1497   %gl993 = load i64, i64* %_printf_prefix_1
1498   %offsetload994 = load i64, i64* %theta987
1499   %plus1995 = add i64 %offsetload994, 1
1500   %intcast996 = ptrtoint i64* %theta987 to i64
1501   %offsetcalc997 = mul i64 %plus1995, 8
1502   %pmath998 = add i64 %intcast996, %offsetcalc997
1503   %pointercast999 = inttoptr i64 %pmath998 to i64*
1504   %rootload1000 = load i64, i64* %pointercast999
1505   %offsetcalc1001 = mul i64 %gl993, 8
1506   %pmath1002 = add i64 %rootload1000, %offsetcalc1001
1507   %pointercast1003 = inttoptr i64 %pmath1002 to i64*
1508   %deref1004 = load i64, i64* %pointercast1003
1509   %derefpntr1005 = inttoptr i64 %deref1004 to i64*
1510   %fpaccess1006 = bitcast i64* %derefpntr1005 to double*
1511   %fpret1007 = load double, double* %fpaccess1006
1512   %printf_ignore1008 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([7 x i8], [7 x i8]*
           @fmt3, i32 0, i32 0), double %fpret1007)
1513   %_printf_prefix_11009 = load i64, i64* %_printf_prefix_1
1514   %tmp1010 = add i64 1, %_printf_prefix_11009
1515   store i64 %tmp1010, i64* %_printf_prefix_1
1516   br label %while991
1517
1518 merge1017:                                        ; preds = %while991
1519   %printf_ignore1018 = call i64 (i8*, ...) @printf(i8* getelementptr inbounds ([2 x i8], [2 x i8]*
           @nlstr, i32 0, i32 0))
1520   %malloc_tmp = tail call i8* @malloc(i32 mul (i32 ptrtoint (i8* getelementptr (i8, i8* null, i32
           1) to i32), i32 50))
1521   %mistakes1020 = load i64, i64* %mistakes
1522   %sprintf_tmp = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp, i8* getelementptr inbounds ([3
           x i8], [3 x i8]* @itosfmt, i32 0, i32 0), i64 %mistakes1020)
1523   %strlen1 = call i32 @strlen(i8* %malloc_tmp)
1524   %strlen2 = call i32 @strlen(i8* getelementptr inbounds ([21 x i8], [21 x i8]* @" mistakes made
           over ", i32 0, i32 0))
1525   %add_tmp = add i32 %strlen1, %strlen2
```

```
  %add_tmp1021 = add i32 %add_tmp, 1
  %mallocsize1022 = mul i32 %add_tmp1021, ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32)
  %malloc_tmp1024 = tail call i8* @malloc(i32 %mallocsize1022)
  %sprintf_tmp1025 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp1024, i8* getelementptr
      inbounds ([5 x i8], [5 x i8]* @concatfmt, i32 0, i32 0), i8* %malloc_tmp, i8* getelementptr
      inbounds ([21 x i8], [21 x i8]* @" mistakes made over ", i32 0, i32 0))
  %malloc_tmp1027 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i8* getelementptr (i8, i8* null,
      i32 1) to i32), i32 50))
  %count1028 = load i64, i64* %count
  %sprintf_tmp1029 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp1027, i8* getelementptr
      inbounds ([3 x i8], [3 x i8]* @itosfmt, i32 0, i32 0), i64 %count1028)
  %strlen11030 = call i32 @strlen(i8* %malloc_tmp1024)
  %strlen21031 = call i32 @strlen(i8* %malloc_tmp1027)
  %add_tmp1032 = add i32 %strlen11030, %strlen21031
  %add_tmp1033 = add i32 %add_tmp1032, 1
  %mallocsize1034 = mul i32 %add_tmp1033, ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32)
  %malloc_tmp1036 = tail call i8* @malloc(i32 %mallocsize1034)
  %sprintf_tmp1037 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp1036, i8* getelementptr
      inbounds ([5 x i8], [5 x i8]* @concatfmt, i32 0, i32 0), i8* %malloc_tmp1024, i8*
      %malloc_tmp1027)
  %strlen11038 = call i32 @strlen(i8* %malloc_tmp1036)
  %strlen21039 = call i32 @strlen(i8* getelementptr inbounds ([14 x i8], [14 x i8]* @"
      iterations.\0A", i32 0, i32 0))
  %add_tmp1040 = add i32 %strlen11038, %strlen21039
  %add_tmp1041 = add i32 %add_tmp1040, 1
  %mallocsize1042 = mul i32 %add_tmp1041, ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32)
  %malloc_tmp1044 = tail call i8* @malloc(i32 %mallocsize1042)
  %sprintf_tmp1045 = call i64 (i8*, i8*, ...) @sprintf(i8* %malloc_tmp1044, i8* getelementptr
      inbounds ([5 x i8], [5 x i8]* @concatfmt, i32 0, i32 0), i8* %malloc_tmp1036, i8*
      getelementptr inbounds ([14 x i8], [14 x i8]* @" iterations.\0A", i32 0, i32 0))
  %printf_ignore1046 = call i64 (i8*, ...) @printf(i8* %malloc_tmp1044)
  ret i64 0
}
```

7.2 The Test Suite

The test suite was written early on in the project in order to establish a standard for our language. However, establishing the test suite this early meant that our language natuturally changed and evolved separately from the standards we had envisioned in the test suite, and tests had to be added, removed, and changed over time.

Our intention is that the final test suite reflects each individual feature we intended to implement. We include multiple test and failure cases for each feature, such as testing tensor operations in multiple dimensions, or testing the different results, uses, and combinations of a single if statement.

We drew from MicroC's example when creating an automated testing script, which runs each test in sequence and takes in its output or its error message, diffing the output file with a premade output file of the output we expected. Then, our script examines the test log to find the number of successful and failing tests–the final test count is 85.

The original test suite was written by Elsbeth Turcan, the tester, and as the project evolved and different team members needed to implement different features, they also edited or contributed particular test cases. Daniel Schwartz proceeded to take over management of the test suite at the end of the project when roles began to mesh.

7.2.1 Tests Included

The test suite tests the following features. The actual code of the test suite, including the test script, is included in Section 9, the Appendix.

- `test-assign.tens`: Tests the assignment of each non-tensor data type, including the multiple ways we can declare floats.
- `test-comments1.tens`: Tests single-line comments being parsed correctly.
- `test-comments2.tens`: Tests multi-line comments being parsed correctly.
- `test-comments3.tens`: Tests that combining single- and multi-line comments works as expected.
- `test-declare.tens`: Tests the declaration of each data type we have, including tensors, and including different types of float notation and combined floats and ints inside tensors.
- `test-div.tens`: Tests element-wise tensor division.
- `test-float-ops1.tens`: Tests printing floats of different formats (e.g., `42.` and `42e2`) as well as signed floats.
- `test-float-ops2.tens`: Tests supported float operations: addition, subtraction, multiplication, division, unary minus, unary plus, greater than, greater than or equal to, less than, and less than or equal to; includes multiple test cases for division and relational operators.
- `test-for1.tens`: Tests a simple for loop that prints the numbers 0 to 4.
- `test-for2.tens`: Tests a for loop implementing a while loop (missing the first and third expressions).
- `test-for3.tens`: Tests nested for loops.
- `test-for4.tens`: Tests a for loop that should never be entered at all (tests that the conditions are evaluated properly and at the right time).
- `test-func1.tens`: Tests calling a simple function and printing its int return value.
- `test-func2.tens`: Tests calling multiple functions, one of which returns a tensor.
- `test-func3.tens`: Tests calling a function with function calls as its arguments.
- `test-func4.tens`: Tests a void function.
- `test-func5.tens`: Tests a recursive function.
- `test-if1.tens`: Tests a simple if.
- `test-if2.tens`: Tests an if-else.
- `test-if3.tens`: Tests an if-else if.
- `test-if4.tens`: Tests an if-else if-else to determine that they chain correctly.
- `test-if5.tens`: Tests an if-else if-else if-else to establish that an arbitrary number of else if's will work.
- `test-if6.tens`: Tests nested ifs.
- `test-if7.tens`: Tests the dangling else problem.
- `test-if8.tens`: Tests nesting ifs along with elses.

- `test-indexpr1.tens`: Tests that expressions used to index into a tensor can be evaluated correctly.
- `test-int-ops.tens`: Tests supported integer operations: addition, subtraction, multiplication, division, unary minus, unary plus, equal to, not equal to, greater than, greater than or equal to, less than, less than or equal to, logical and, logical or, and logical not; includes multiple test cases for division and relational operators.
- `test-loops1.tens`: Tests some simple for loops with while loops inside.
- `test-loops2.tens`: Tests a simple while loop with a for loop inside changing the flag that exits the loop.
- `test-promotion.tens`: Tests the automatic promotion of integers to floats when performing integer/float operations.
- `test-scope1.tens`: Tests our static scoping rules.
- `test-slice.tens`: Tests tensor slicing, using some tensor immediates as well as tensor multiplication.
- `test-sqrt.tens`: Tests the library function `sqrt`.
- `test-stdlib1-pow.tens`: Tests the library function `sqrt`.
- `test-stdlib2-max.tens`: Tests the library function `max`.
- `test-stdlib3-min.tens`: Tests the library function `min`.
- `test-stdlib4-abs.tens`: Tests the library function `abs`.
- `test-string-ops.tens`: Tests string concatenation.
- `test-strlen.tens`: Tests the library function `strlen`.
- `test-tenexpr1.tens`: Tests that the types of expressions that are allowed in a tensor immediate are really allowed and evaluated properly.
- `test-tensoraccess.tens`: Tests accessing tensors by specifying the indices.
- `test-tensor-add.tens`: Tests tensor addition.
- `test-tensor-add2.tens`: Tests tensor addition with tensors of different rank.
- `test-tensordotproduct.tens`: Tests dot product, a special case of tensor multiplication when both tensors are of rank 1 with one bound index.
- `test-tensorimmed.tens`: Tests declaring and accessing tensor immediates.
- `test-tensor-sub.tens`: Tests tensor subtraction.
- `test-tenor-sub2.tens`: Tests tensor subtraction with tensors of different rank.
- `test-tmult1.tens`: Tests tensr multiplication.
- `test-tmult2.tens`: Tests more complicated tensor multiplication, including variables inside tensor immediates.
- `test-tmult3.tens`: Continues testing tensor multiplication.
- `test-while1.tens`: Tests a simple while loop that prints the numbers from 0 to 4.
- `test-while2.tens`: Tests a while loop with a generic integer as its boolean condition.

- `fail-add.tens`: Tests improper types in addition.
- `fail-and.tens`: Tests improper types in logical and.
- `fail-assign1.tens`: Tests assigning an int to a float, and then an actual tye mismatch in an assignment.
- `fail-assign2.tens`: Tests trying to use a statement on the right-hand side of an assignment.
- `fail-codeoutsidefunc.tens`: Tests trying to place statements outside of a function, which is not allowed.
- `fail-comments.tens`: Tests improper comment syntax.
- `fail-declare.tens`: Tests illegal characters in an identifier name.
- `fail-declare2.tens`: Tests trying to ue a reserved word as an identifier name.
- `fail-declare3.tens`: Tests trying to declare a string with invalid quotes.
- `fail-declare4.tens`: Tests redefinition of an identifier within the same block.
- `fail-declare-tensor1.tens`: Tests trying to place a string literal inside a tensor immediate.
- `fail-declare-tensor2.tens`: Tests trying to declare a tensor using a tensor immediate whose internal dimensions are not consistent.
- `fail-expr.tens`: Tests trying to use a poorly-formatted expression (a syntax error).
- `fail-func1.tens`: Tests attempting to nest function definitions.
- `fail-func2.tens`: Tests trying to overload functions. We were close to implementing function overloading and so would have changed this test!
- `fail-func3.tens`: Tests a missing return statement in a function that should have one according to its type.
- `fail-func4.tens`: Tests a function that returns a different type than it was declared to return.
- `fail-func5.tens`: Tests trying to return something from a void function.
- `fail-geq.tens`: Tests improper types in greater-than-or-equal-to.
- `fail-gt.tens`: Tests improper types in greater-than.
- `fail-leq.tens`: Tests improper types in less-than-or-equal-to.
- `fail-lt`: Tests improper types in less-than.
- `fail-nobraces.tens`: Tests mismatched braces in control flow.
- `fail-nomain.tens`: Tests a .tens file without a main method.
- `fail-noparens.tens`: Tests mismatched parentheses in expressions.
- `fail-nosemi.tens`: Tests missing semicolon.
- `fail-or.tens`: Tests improper types in logical or.
- `fail-strcat.tens`: Tests improper types in string concatenation.
- `fail-sub.tens`: Tests improper types in subtraction.

- `fail-tensor-add.tens`: Tests improper types in tensor addition (i.e., tensors of different rank).
- `fail-tensor-negate.tens`: Tests improper types (i.e., a tensor) in unary minus.
- `fail-tensor-sub2.tens`: Tests improper types (i.e., tensors of different rank) in tensor subtraction.
- `fail-uminus.tens`: Tests improper types (string) in unary minus.

7.2.2 Running the Test Suite

The test suite and success/fail counter script are shown executed below:

```
plt4115@plt4115:~/Desktop/LaTenS$ ./testall.sh
test-assign...OK
test-comments1...OK
test-comments2...OK
test-comments3...OK
test-declare...OK
test-div...OK
test-float-ops1...OK
test-float-ops2...OK
test-for1...OK
test-for2...OK
test-for3...OK
test-for4...OK
test-func1...OK
test-func2...OK
test-func3...OK
test-func4...OK
test-func5...OK
test-if1...OK
test-if2...OK
test-if3...OK
test-if4...OK
test-if5...OK
test-if6...OK
test-if7...OK
test-if8...OK
test-indexpr1...OK
test-int-ops...OK
test-loops1...OK
test-loops2...OK
test-promotion...OK
test-scope1...OK
test-slice...OK
test-sqrt...OK
test-stdlib1-pow...OK
test-stdlib2-max...OK
test-stdlib3-min...OK
test-stdlib4-abs...OK
test-string-ops...OK
test-strlen...OK
test-tenexpr1...OK
testor-add.tens...OK
testor-add2.tens...OK
testor-dotproduct.tens...OK
testor-sub.tens...OK
```

```
testor-sub2.tens...OK
testoraccess.tens...OK
testorimmed.tens...OK
test-tmult1...OK
test-tmult2...OK
test-tmult3...OK
test-while1...OK
test-while2...OK
fail-add...OK
fail-and...OK
fail-assign1...OK
fail-assign2...OK
fail-codeoutsidefunc...OK
fail-comments...OK
fail-declareor1.tens...OK
fail-declareor2.tens...OK
fail-declare1...OK
fail-declare2...OK
fail-declare3...OK
fail-declare4...OK
fail-expr...OK
fail-func1...OK
fail-func2...OK
fail-func3...OK
fail-func4...OK
fail-func5...OK
fail-geq...OK
fail-gt...OK
fail-leq...OK
fail-lt...OK
fail-nobraces...OK
fail-nomain...OK
fail-noparens...OK
fail-nosemi...OK
fail-or...OK
fail-strcat...OK
fail-sub...OK
failor-add.tens...OK
failor-negate.tens...OK
failor-sub2.tens...OK
fail-uminus...OK
plt4115@plt4115:~/Desktop/LaTenS$ ./script_num_pass.sh
     85 SUCCESS
```

---

8 Lessons Learned

8.1 Daniel Schwartz

- .gitignore! Use it! It will prevent you from committing auto-generated files and make managing your git repo in general much easier. Also commit -am is a great shortcut. On the topic of github, NEVER EVER commit broken code.
- Error handling. It seems least important at first but the earlier you get this infrastructure in, the easier it is to troubleshoot and therefore the easier it is to move forward in general. One of the best

things I accomplished was adding a line number to every error message in our semantic checker. We went from averaging 1+ minutes to fix a single mistake in our test/demo .tens files to an average of 5-10 seconds. Once you have 80+ tests and 3 longer demos, you can see how this helps.

- The value of making sure every teammate's environment is as close to identical as reasonably possible. Although we used a mix of operating systems, we tried to sync our OCaml and llvm versions. Even so, we ran into some problems compiling each other's code.
- Some of us worked best in marathon sessions started in the evening and lasting until late at night once a week. Others preferred to code solo and on their own schedule. As a manager, I learned the importance of balancing people's preferences with what the group may need at the moment and motivating people to do what was needed. This can also be extrapolated to cover the importance of giving people features they are particularly equipped to implement or would particularly enjoy but at the same time, making sure people who are needed in one place aren't busy in another, less-important place, even if it means a little less fun at the moment.
- Have fun :) Managing was fun for me. Working with a team was fun and not something you get to do in so many CS classes. You're also given a lot of creative freedom and very sporadic assignments in this class. So enjoy yourself.

8.2 Eliana Ward-Lev

- You have to understand something much better to write code to do it than to simply do it yourself
- Learning how to code in a language is something done best with practice. The earlier this practice occurs, the earlier you can stop fighting the language and start using it.
- Lots of checking in the parser isn't a good thing. Parser errors are harder to make useful and it requires more duplication. Semant is well suited for semantic checking and it will be much easier to write any and all checks there.
- Collaborating with people isn't always easy and fun. The sooner you accept the fact that people sometimes have clashing styles the happier you will be. Collaborating with people however can also be really useful. Someone else can almost always find an error faster than you can as they haven't been staring at it quite as long.
- When you get an error in a test first check that the test was written correctly before assuming the compiler was the problem.

8.3 Elsbeth Turcan

- It's a lot easier to coordinate and implement features if you sit down with your team members and code together in the same room. It's not the ideal environment for everyone, as everyone has their own preferred way to code, but it does help to be able to ask questions about parts of the code that aren't yours, or about frustrating issues that you run into.
- It helps to get into the thick of the coding as early as possible. Planning out architectures and pipelines and features is important, but there's no substitute for experience. Actually sitting down and implementing the skeletons of your code so that you can fill them in later will be much more helpful in understanding how to finish the project than planning endlessly. Particularly because you are not likely to have much experience programming in OCaml before this project–the more you code your compiler in OCaml, the more you'll understand.

- Standardize your syntax early and make sure you and your team members know it. There's nothing more frustrating than trying to figure out why your test programs are breaking and figuring out twenty minutes later that you forgot a semicolon, or the word "let". Early implementation of good error handling also helps.
- Don't do too much error checking in your parser. It's horrendous to try and handle errors gracefully in your parser. Horrendous.
- LLVM is all about cute foxes.

8.4 Mohit Rajpal

- LLVM's strong type checking is a nuisance when generating code programmatically.
- Hacking together a code generation backend is perfectly feasible in under a week.
- My time estimates are often very low end of actual time requirements.
- x86_64 assembly is easier to read than LLVM-IR.
- Tensor multiplication syntax and implications.
- I thought this class would be easy since I already knew the material but I was wrong.

9 Appendix

9.1 .gitignore

```
*.ll
*.diff
*~
*.err
*.out
!tests/*.err
!tests/*.out
test.1
test_input.tens
```

9.2 Makefile

```
# Make sure ocamlbuild can find opam-managed packages: first run
#
# eval `opam config env`

# Easiest way to build: using ocamlbuild, which in turn uses ocamlfind

.PHONY : latens.native

latens.native :
   ocamlbuild -use-ocamlfind -pkgs llvm,llvm.analysis -cflags -w,+a-4-5-8-10-11-26-27-33 -tag
       debug \
      latens.native
```

```
# "make clean" removes all generated files

.PHONY : clean
clean :
   ocamlbuild -clean
   rm -rf testall.log *.diff latens scanner.ml parser.ml parser.mli
   rm -rf parser.conflicts parser.automaton parser.output
   rm -rf *.cmx *.cmi *.cmo *.cmx *.o *~ *.out tests/*~
   rm -rf out.* *.ll *.err
   rm -rf intermediate.tens

# More detailed: build using ocamlc/ocamlopt + ocamlfind to locate LLVM

OBJS = exceptions.cmx ast.cmx sast.cmx parser.cmx scanner.cmx semant.cmx latens.cmx
#OBJS = exceptions.cmx ast.cmx sast.cmx codegen.cmx parser.cmx scanner.cmx semant.cmx latens.cmx


latens : $(OBJS)
   ocamlfind ocamlopt -linkpkg -package llvm -package llvm.analysis $(OBJS) -o latens

scanner.ml : scanner.mll
   ocamllex scanner.mll

parser.ml parser.mli : parser.mly
   ocamlyacc parser.mly

%.cmo : %.ml
   ocamlc -c $<

%.cmi : %.mli
   ocamlc -c $<

%.cmx : %.ml
   ocamlfind ocamlopt -c -package llvm $<

### Generated by "ocamldep *.ml *.mli" after building scanner.ml and parser.ml
ast.cmo :
ast.cmx :
sast.cmo : ast.cmo
sast.cmx : ast.cmx
exceptions.cmo :
exceptions.cmx :
codegen.cmo : ast.cmo sast.cmo
codegen.cmx : ast.cmx sast.cmx
latens.cmo : exceptions.cmo sast.cmo semant.cmo scanner.cmo parser.cmi codegen.cmo ast.cmo
latens.cmx : exceptions.cmx sast.cmx semant.cmx scanner.cmx parser.cmx codegen.cmx ast.cmx
parser.cmo : ast.cmo parser.cmi
parser.cmx : ast.cmx parser.cmi
scanner.cmo : parser.cmi
scanner.cmx : parser.cmx
semant.cmo : ast.cmo sast.cmo
semant.cmx : ast.cmx sast.cmx
parser.cmi : ast.cmo

# Building the tarball
```

```
TESTS = add1 arith1 arith2 arith3 fib for1 for2 func1 func2 func3 \
    func4 func5 func6 func7 func8 gcd2 gcd global1 global2 global3 \
    hello if1 if2 if3 if4 if5 local1 local2 ops1 ops2 var1 var2  \
    while1 while2

FAILS = assign1 assign2 assign3 dead1 dead2 expr1 expr2 for1 for2 \
    for3 for4 for5 func1 func2 func3 func4 func5 func6 func7 func8 \
    func9 global1 global2 if1 if2 if3 nomain return1 return2 while1 \
    while2

TESTFILES = $(TESTS:%=test-%.mc) $(TESTS:%=test-%.out) \
      $(FAILS:%=fail-%.mc) $(FAILS:%=fail-%.err)


TARFILES = exceptions.ml ast.ml sast.ml Makefile latens.ml parser.mly README scanner.mll \
#TARFILES = exceptions.ml ast.ml sast.ml codegen.ml Makefile latens.ml parser.mly README
    scanner.mll \
   semant.ml testall.sh $(TESTFILES:%=tests/%)

latens-llvm.tar.gz : $(TARFILES)
   cd .. && tar czf latens-llvm/latens-llvm.tar.gz \
     $(TARFILES:%=latens-llvm/%)
```

9.3 README.md

```
# LaTenS

| Daniel Schwartz | Eliana Ward-Lev | Elsbeth Turcan | Mohit Rajpal |
|:---------------:|:---------------:|:--------------:|:------------------:|
|    `ds3263`     |    `erw2138`    |    `ect2150`   |     `mr3522`       |
|    *Manager*    |  *Language Guru*|    *Tester*    | *System Architect* |


To run code:
0) ./latens.native -s < test_input.tens > out.ll
   - does semantic checking on test_input.tens and pretty prints sast into out.ll
   - -a flag for printing ast
1) /usr/local/opt/llvm/bin/lli out.ll
   - runs a .ll file

~~~~~ Links ~~~~~
1) https://www.sharelatex.com/project/57e067ad0934347f40c1dea4
2) http://llvm.org/releases/3.8.0/docs/tutorial/OCamlLangImpl3.html
3) https://llvm.moe/ocaml/
4) http://llvm.org/releases/2.6/docs/LangRef.html

Commenting:
for latex printing: only allowed before blocks and fdecls: can be any number of lines beginning
    with %%.
    - note: the block/fdecl must continue on the next line
    - similarly, the first %% must be on its own line

Scripts:
```

```
1) script_num_pass.sh counts the number of tests that pass and the number that fail. Divide the
    number that fail by two.
```

9.4 latens.ml

```
(* Top-level of the LaTenS compiler: scan & parse the input,
   check the resulting AST, generate LLVM IR, and dump the module *)
(*open Sast*)
open Exceptions
open Printf

type action = Ast | Past | Sast | Debug | LLVM_IR | Compile


let _ =
  let med = "intermediate.tens" in
  let file = "stdlib.tens" in
  let in_channel = try (if Array.length Sys.argv > 1 && Array.length Sys.argv < 4 then (open_in
      Sys.argv.(Array.length Sys.argv - 1)) else
       (if Array.length Sys.argv == 1 then stdin else raise (ImproperArgsException)))
       with ImproperArgsException -> prerr_endline("Usage:\n./latens.ml <compiler flag>
           <file>\n./latens.ml <file>"); exit 1
    in let out_channel = open_out med in
      (try
        while true do
          let line = input_line in_channel in
          fprintf out_channel "%s\n" line
        done
      with End_of_file ->
        close_in in_channel);
          let in_channel = open_in file
      in (try
        while true do
          let line = input_line in_channel in
          fprintf out_channel "%s\n" line
        done
      with End_of_file ->
        close_in in_channel);
      close_out out_channel;

      let in_file = open_in med in


  let action = if Array.length Sys.argv > 1 then
    try List.assoc Sys.argv.(1) [ ("-a", Ast);(* Print the AST only *)
      ("-p", Past);(* Pretty Print only *)
      ("-s", Sast); (* Semantically check and print Sast *)
      ("-d", Debug); (* Pretty print the Sast and also the final global environment *)
      ("-l", LLVM_IR); (* Generate LLVM, don't check *)
      ("-c", Compile);] (* Generate, check LLVM IR *)
    with Not_found -> Compile
  else Compile in
  (try
    let lexbuf = Lexing.from_channel in_file in
```

```
    let ast = Parser.program Scanner.token lexbuf in
    (*let sast = Semant.typify(ast) in*)
  match action with
    Ast -> print_string (Ast.string_of_program ast)
  | Past -> print_string (Ast.pretty_program ast)
  | Sast -> print_string (Sast.string_of_program (snd (Semant.typify ast)))
  | Debug -> print_string (Semant.string_of_prog_and_env (Semant.typify ast))
  | LLVM_IR -> print_string (Llvm.string_of_llmodule (Codegen.translate (snd (Semant.typify
      ast))))
  | Compile -> let m = Codegen.translate (snd (Semant.typify ast)) in
    Llvm_analysis.assert_valid_module m;
    print_string (Llvm.string_of_llmodule m)
with UnmatchedQuotationException(line) -> prerr_endline ("Line: " ^ string_of_int line ^ ":
    Unmatched quotation"); close_in in_file; Sys.remove med; exit 1
    | IllegalCharacterException(line,c) -> prerr_endline ("Line: "^ string_of_int line ^ ":
        Illegal character: " ^ Char.escaped c); close_in in_file; Sys.remove med; exit 1
    | ParserException(sline, s, eline, e, msg) -> prerr_endline ("Line: " ^ string_of_int sline ^
        ": Char " ^ string_of_int s ^ " - Line: " ^ string_of_int eline ^ " Char " ^
        string_of_int e ^ ": " ^ msg); close_in in_file; Sys.remove med; exit 1
    | NoExprHasNoLineNumberException -> prerr_endline ("[Debug message] You modified the
        compiler. Something is wrong. Why do you want the linenum of a noexpr"); close_in
        in_file; Sys.remove med; exit 1
    | Parsing.Parse_error -> prerr_endline " shouldn't happen "; close_in in_file; Sys.remove
        med; exit 1
    | NoMainException -> prerr_endline "LaTenS files need a main method."; close_in in_file;
        Sys.remove med; exit 1
    | StatementAfterReturnException(exp) -> prerr_endline ("Found statement: " ^ exp ^ " after
        return."); close_in in_file; Sys.remove med; exit 1
    | ExpectedConditionException(line, e) -> prerr_endline ("Line: " ^ line ^ ": Conditional
        expression expected, but " ^ e ^ " does not evaluate to int."); close_in in_file;
        Sys.remove med; exit 1
    | BinaryTypeMismatchException(line, t1,t2, op) -> prerr_endline ("Line: " ^ line ^ ":
        Operator " ^ op ^ " does not support types " ^ t1 ^ " and " ^ t2 ^ "."); close_in
        in_file; Sys.remove med; exit 1
    | UnaryTypeMismatchException(line, t, op) -> prerr_endline ("Line: " ^ line ^ ": Type " ^ t ^
        " not supported by op: " ^ op ^ "."); close_in in_file; Sys.remove med; exit 1
    | BadConcatException(line) -> prerr_endline ("Line: " ^ line ^ ": Concatenation can only be
        used with strings."); close_in in_file; Sys.remove med; exit 1
    | NotDeclaredInScopeException(line, id) -> prerr_endline ("Line: " ^ line ^ ": " ^ id ^ " not
        declared in current scope."); close_in in_file; Sys.remove med; exit 1
    | DoubleDeclareException(line, id) -> prerr_endline ("Line: " ^ line ^ ": " ^ id ^ " already
        declared in current scope."); close_in in_file; Sys.remove med; exit 1
    | FuncReturnTypeException(line, t1, t2) -> prerr_endline ("Line: " ^ line ^ ": Expected
        return type of " ^ t1 ^ " but found " ^ t2 ^ " instead."); close_in in_file; Sys.remove
        med; exit 1
    | InvalidTensorImmedException(line) -> prerr_endline("Line: " ^ line ^ ": Invalid tensor
        immediate."); close_in in_file; Sys.remove med; exit 1
    | InvalidIndexException(line, t) -> prerr_endline ("Line: " ^ line ^ ": Int expected for
        tensor index. Found: " ^ t ^ "."); close_in in_file; Sys.remove med; exit 1
    | TensorTypeException(line, s, t1, t2) -> prerr_endline ("Line: " ^ line ^ ": " ^ s ^ " was
        declared as " ^ t1 ^ " but you used " ^ t2 ^ " indices."); close_in in_file; Sys.remove
        med; exit 1
    | InvalidTensorElementException(line, t) -> prerr_endline ("Line: " ^ line ^ ": Type " ^ t ^
        " is not a valid tensor element."); close_in in_file; Sys.remove med; exit 1
    | EmptyTensorException(line) -> prerr_endline("Line: " ^ line ^ ": Cannot declare a
        0-dimensional tensor."); close_in in_file; Sys.remove med; exit 1
```

```
    | VoidTensorIndexException(line, expr) -> prerr_endline("Line: " ^ line ^ ": " ^ expr ^ " is
        void in the current scope and is not a valid tensor index."); close_in in_file;
        Sys.remove med; exit 1
    | UnspecifiedTensorMultiplicationException(line) -> prerr_endline("Line: " ^ line ^ ": Cannot
        multiply tensors with no specified indices."); close_in in_file; Sys.remove med; exit 1
    | NoReturnException(t) -> prerr_endline("Expected return statement of type: " ^ t ^ " but no
        return statement found."); close_in in_file; Sys.remove med; exit 1
    | CodegenException -> prerr_endline("Codegen exception. Gah"); close_in in_file; Sys.remove
        med; exit 1);
 close_in in_file; Sys.remove med; exit 0
```

9.5 exceptions.ml

```
(* Scanner *)
exception IllegalCharacterException of int * char
exception UnmatchedQuotationException of int
exception IllegalLatexComment of int * string


(* Parser *)
exception ParserException of int * int * int * int * string


(* Ast *)
exception NoExprHasNoLineNumberException
exception CallbackExprInAst


(* Semant *)
exception NoMainException
exception StatementAfterReturnException of string
exception ExpectedConditionException of string * string
exception BinaryTypeMismatchException of string * string * string * string
exception UnaryTypeMismatchException of string * string * string
exception BadConcatException of string
exception NotDeclaredInScopeException of string * string
exception DoubleDeclareException of string * string
exception FuncReturnTypeException of string * string * string
exception NoReturnException of string


(* Tensors *)
exception InvalidIndexException of string * string
exception InvalidTensorImmedException of string
exception InvalidTensorElementException of string * string
exception TensorTypeException of string * string * string * string
exception EmptyTensorException of string
exception VoidTensorIndexException of string * string
exception UnspecifiedTensorMultiplicationException of string

(* Codegen *)
exception CodegenException

(* Latens *)
```

```
exception ImproperArgsException
```

9.6 ast.ml

```
type op = Add | Sub | Mult | Div | Equal | Neq |
        Less | Leq | Greater | Geq | And | Or | Concat
type uop = Neg | Not

type line = string

type comment = (string * string) list

type typ = Void | Float | Int | String | TensorType of qualifier_rettype
and
qualifier_rettype = expr list
and
accessor = IntIndex of int | ExprIndex of expr
and
bind = InferredBind of line * string * expr
| TensorBind of line * string * qualifier_rettype
| TensorBindAndAssign of line * string * qualifier_rettype * expr
and
expr =
  Id of line * string               | Noexpr
| Binop of line * expr * op * expr  | Unop of line * uop * expr
| Assign of line * string * expr    | Call of line * string * expr list
| Iliteral of line * int
| Fliteral of line * float
| Sliteral of line * string
| TensorId of line * string * qualifier_rettype
| TensorAssign of line * string * qualifier_rettype * expr
| Tensor of line * expr list
| BindExpr of line * bind


type stmt = Block of comment * stmt list
| Expr of comment * expr
| If of comment * expr * stmt * stmt
| For of comment * expr * expr * expr * stmt
| While of comment * expr * stmt
| Return of comment * expr



type func_decl = {
  comm : (line * string) list;
   typ : typ;
   fname : string;
   formals : (typ * string) list;
   body : stmt
}

type program = bind list * func_decl list
```

```
let get_line_expr = function
    Iliteral(l,_) -> l
  | Fliteral(l,_) -> l
  | Sliteral(l,_) -> l
  | Binop(l,_,_,_) -> l
  | Unop(l,_,_) -> l
  | Id(l,_) -> l
  | TensorId(l,_,_) -> l
  | TensorAssign(l,_,_,_) -> l
  | Tensor(l,_) -> l
  | Assign(l,_,_) -> l
  | Call(l,_,_) -> l
  | BindExpr(l,_) -> l
  | Noexpr -> raise( Exceptions.NoExprHasNoLineNumberException)

let get_line_bind = function
    InferredBind(l,_,_) -> l
  | TensorBind(l,_,_) -> l
  | TensorBindAndAssign(l,_,_,_) -> l

let string_of_op = function
    Add -> "+"
  | Sub -> "-"
  | Mult -> "*"
  | Div -> "/"
  | Equal -> "=="
  | Neq -> "~="
  | Less -> "<"
  | Leq -> "<="
  | Greater -> ">"
  | Geq -> ">="
  | And -> "&&"
  | Or -> "||"
  | Concat -> "^"

let string_of_uop = function
    Neg -> "-"
  | Not -> "~"

let string_of_qual_list q = String.concat ", " q

let rec string_of_bind = function
    InferredBind(_, s, e) -> "let " ^ s ^ " = " ^ string_of_expr e
  | TensorBind(_, s, q) -> "let " ^ s ^ "_{" ^ string_of_qual_list_expr q ^ "}"
  | TensorBindAndAssign(_, s, q, e) -> "let " ^ s ^ "_{" ^ string_of_qual_list_expr q ^ "} = " ^
       string_of_expr e


and string_of_qual_list_expr q = String.concat ", " (List.map string_of_expr q)

and string_of_expr = function
    Id(_,s) -> s
  | Binop(_, e1, op, e2) -> string_of_expr e1 ^ " " ^ string_of_op op ^ " " ^ string_of_expr e2
  | Unop(_,uop, e) -> string_of_uop uop ^ " " ^ string_of_expr e
  | Assign(_,var, e) -> var ^ " = " ^ string_of_expr e
  | Iliteral(_,i) -> string_of_int i
```

```
  | Sliteral(_,s) -> s
  | Fliteral(_,f) -> string_of_float f
  | BindExpr(_,b) -> string_of_bind b
  | Noexpr -> ""
  | Call(_,fn, el) -> fn ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | TensorId(_, s, q) -> s ^ "_{" ^ string_of_qual_list_expr q ^ "}"
  | TensorAssign(_, s, sl, e) -> s ^ "_{" ^ string_of_qual_list_expr sl ^
                          "} = " ^ string_of_expr e
  | Tensor(_, t) -> "[" ^ String.concat ", " (List.map string_of_expr t) ^ "]"

(*and string_of_tensor = function
   |ExprDim(el) -> "[" ^ String.concat ", " (List.map string_of_expr el) ^ "]"*)

let rec string_of_stmt = function
    Block(_, stmts) -> "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
  | Expr(_, e) -> string_of_expr e ^ ";\n"
  | Return(_, e) -> "return " ^ string_of_expr e ^ ";\n"
  | If(_, e, s, Block(_, [])) -> "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s
  | If(_, e, s1, s2) -> "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s1 ^ " else\n" ^
      string_of_stmt s2
  | For(_, e1, e2, e3, s) -> "for (" ^ string_of_expr e1 ^ " ; " ^ string_of_expr e2 ^ " ; " ^
      string_of_expr e3 ^ ") " ^ string_of_stmt s
  | While(_, e, s) -> "while (" ^ string_of_expr e ^ ") " ^ string_of_stmt s


let string_of_typ = function
    Void -> "void"
  | Float -> "float"
  | Int -> "int"
  | String -> "string"
  | TensorType(q) -> string_of_int (List.length q)

let string_of_fdecl fdecl =
  string_of_typ fdecl.typ ^ " " ^
  fdecl.fname ^ "(" ^ String.concat ", " (List.map snd fdecl.formals) ^
  ")\n" ^
  string_of_stmt fdecl.body

let string_of_program (globals, functions) =
  String.concat "\n" (List.map string_of_bind globals) ^ "\n" ^
  String.concat "\n" (List.map string_of_fdecl functions)




(* Pretty Printing for LaTeX file generation*)
let rec pretty_print_binop (e1, op, e2) = match op with
    Add -> "(" ^ pretty_print_expr e1 ^ "+" ^ pretty_print_expr e2 ^ ")"
  | Sub -> "(" ^ pretty_print_expr e1 ^ "-" ^ pretty_print_expr e2 ^ ")"
  | Mult -> pretty_print_expr e1 ^ "\\cdot " ^ pretty_print_expr e2
  | Div -> "\\frac {" ^ pretty_print_expr e1 ^ "} {" ^ pretty_print_expr e2 ^ "}"
  | Equal -> pretty_print_expr e1 ^ "==" ^ pretty_print_expr e2
  | Neq -> pretty_print_expr e1 ^ "\\neq" ^ pretty_print_expr e2
  | Less -> pretty_print_expr e1 ^ "<" ^ pretty_print_expr e2
  | Leq -> pretty_print_expr e1 ^ "\\leq" ^ pretty_print_expr e2
```

```
  | Greater -> pretty_print_expr e1 ^ ">" ^ pretty_print_expr e2
  | Geq -> pretty_print_expr e1 ^ "\\geq" ^ pretty_print_expr e2
  | And -> pretty_print_expr e1 ^ "\\& \\&" ^ pretty_print_expr e2
  | Or -> pretty_print_expr e1 ^ "||" ^ pretty_print_expr e2
  | Concat -> pretty_print_expr e1 ^ "\\hbox{ \\string^ }" ^ pretty_print_expr e2

and pretty_print_uop = function
    Neg -> "-"
  | Not -> "\\textasciitilde"

and pretty_print_indices q = String.concat ", " (List.map pretty_print_expr q)

and pretty_print_bind = function
    InferredBind(_, s, e) -> "\\hbox{\\textcolor{violet}{\\bf let} \\textcolor{BurntOrange}{" ^ s
        ^ "} = }" ^ pretty_print_expr e
  | TensorBind(_, s, q) -> "\\hbox{\\textcolor{violet}{\\bf let} " ^ s ^ "}_{" ^
      pretty_print_indices q ^ "}"
  | TensorBindAndAssign(_, s, q, e) ->
      "\\hbox{\\textcolor{violet}{\\bf let} " ^ s ^ "}_{" ^ pretty_print_indices q ^ "} = " ^
          pretty_print_expr e

and pretty_print_comment = function
    [] -> ""
  | hd :: tl when tl != [] -> "\\hbox{\\textcolor{Purple}{" ^ snd hd ^ "}}" ^ "\\\\\n" ^
      pretty_print_comment tl
  | hd :: tl -> "\\hbox{\\textcolor{Purple}{" ^ snd hd ^ "}}" ^ "\n" ^ pretty_print_comment tl

and pretty_print_expr = function
    Id(_, s) -> "\\hbox{" ^ s ^ "} "
  | Binop(_, e1, op, e2) -> pretty_print_binop (e1, op, e2)
  | Unop(_, uop, e) -> "(" ^ pretty_print_uop uop ^ pretty_print_expr e ^ ")"
  | Assign(_, var, e) -> var ^ " = " ^ pretty_print_expr e
  | Iliteral(_, i) -> string_of_int i
  | Sliteral(_, s) -> "\\hbox{\\textcolor{LimeGreen}{``" ^ s ^ "\"}}"
  | Fliteral(_, f) -> string_of_float f
  | BindExpr(_, b) -> pretty_print_bind b
  | Noexpr -> ""
  | Call(_,fn, el) -> "\\hbox{" ^ fn ^ "}(" ^
                String.concat ", " (List.map pretty_print_expr el) ^ ")"
  | TensorId(_, s, q) -> s ^ "_{" ^ pretty_print_indices q ^ "}"
  | TensorAssign(_, s, sl, e) -> s ^ "}_{" ^ pretty_print_indices sl ^ "} = " ^ pretty_print_expr e
  | Tensor(_, t) -> pretty_print_tensor t

and pretty_print_line l = String.concat " & " (List.map pretty_print_expr l)

and pretty_print_matrix m = String.concat " \\\\ " (List.map (fun n -> (match n with Tensor(_, t)
    -> pretty_print_line t | _ -> "")) m)

and pretty_print_tensor t = match List.hd t with
    Tensor(_, st) -> (match List.hd st with
        Tensor(_, _) -> "\\left[" ^ String.concat ", " (List.map pretty_print_btensor t) ^
            "\\right]"
       |_ -> "\\begin{pmatrix} " ^ pretty_print_matrix t ^ "\\end{pmatrix}")
   | _ -> "\\begin{pmatrix} " ^ String.concat " & " (List.map string_of_expr t) ^ "\\end{pmatrix}"

and pretty_print_btensor t = match t with
```

```
    Tensor(_, st) -> "\\left[" ^ String.concat ", " (List.map pretty_print_btensor st)
 ^ "\\right]"
   |_ -> pretty_print_expr t


let rec pretty_print_stmt = function
    Block(comm, stmts) -> pretty_print_comment comm ^ "\\\\" ^
                  "\\{ \n \\begin{adjustwidth}{2.5em}{0pt} \n" ^
                 String.concat "" (List.map pretty_print_stmt stmts) ^
                 "\\end{adjustwidth} \\} \\\\ \n"
  | Expr(comm, e) -> pretty_print_comment comm ^ "\\\\" ^
                  "$" ^ pretty_print_expr e ^ "$;\\\\ \n"
  | Return(comm, e) -> pretty_print_comment comm ^ "\\\\" ^
                "\\textcolor{Aquamarine}{\\bf return} $" ^
                pretty_print_expr e ^ "$;\n"
  | If(comm, e, s, Block(_, [])) -> pretty_print_comment comm ^ "\\\\" ^
                        "if ($" ^ pretty_print_expr e ^ "$)" ^
                         pretty_print_stmt s
  | If(comm, e, s1, s2) -> pretty_print_comment comm ^ "\\\\" ^
                   "if ($" ^ pretty_print_expr e ^ "$) " ^
                   pretty_print_stmt s1 ^ " else " ^ pretty_print_stmt s2
  | For(comm, e1, e2, e3, s) -> pretty_print_comment comm ^ "\\\\" ^
                        "for ($" ^ pretty_print_expr e1 ^ "$ ; $" ^
                        pretty_print_expr e2 ^ "$ ; $" ^ pretty_print_expr e3 ^
                        "$) " ^ pretty_print_stmt s
  | While(comm, e, s) -> pretty_print_comment comm ^ "\\\\" ^
                      "while ($" ^ pretty_print_expr e ^ "$) " ^ pretty_print_stmt s

let pretty_print_typ = function
    Void -> "void"
  | Float -> "float"
  | Int -> "int"
  | String -> "string"
  | TensorType(q) -> string_of_int (List.length q) ^ "-tensor"

let pretty_print_fdecl fdecl =
  pretty_print_comment fdecl.comm ^ "\n" ^
  "\\textcolor{blue}{\\bf " ^ pretty_print_typ fdecl.typ ^ "} " ^
  "\\textcolor{BlueViolet}{ \\bf " ^ fdecl.fname ^ "}(" ^ String.concat ", " (List.map snd
      fdecl.formals) ^
  ") \n" ^
  pretty_print_stmt fdecl.body

let pretty_print_glob g = "$" ^ pretty_print_bind g ^ "$;\\\\ \n"

let pretty_program (globals, functions) =
     "\\documentclass{article} \n" ^
     "\\usepackage{fullpage} \n" ^
     "\\usepackage{listings} \n" ^
     "\\usepackage{color} \n" ^
     "\\usepackage{amsmath} \n" ^
     "\\usepackage[dvipsnames]{xcolor} \n" ^
     "\\usepackage{changepage} \n" ^
     "\\parindent 0pt \n" ^
     "\\renewcommand{\\baselinestretch}{1.0} \n \n " ^
     "\\setlength\\delimitershortfall{0pt} \n" ^
```

```
    "\\setlength\\delimiterfactor{1200} \n \n" ^
    "\\begin{document} \n" ^
    String.concat "" (List.map pretty_print_glob globals) ^
    String.concat "\\\\ \n" (List.map pretty_print_fdecl functions) ^
    "\\end{document}"
```

9.7 codegen.ml

```
1
2  (* Code generation: translate takes a semantically checked AST and
3  produces LLVM IR
4
5  LLVM tutorial: Make sure to read the OCaml version of the tutorial
6
7  http://llvm.org/docs/tutorial/index.html
8
9  Detailed documentation on the OCaml LLVM library:
10
11 http://llvm.moe/
12 http://llvm.moe/ocaml/
13
14 *)
15
16 exception TODO of string
17 exception COMPILE_ERROR of string
18
19 module L = Llvm
20 module A = Ast
21 module S = Sast
22 module Sm = Semant
23 open Stack
24
25
26 type 'a vi = Free of 'a | Bound of 'a
27 type tensordim = Recurse of L.llvalue * (tensordim list) | Base of L.llvalue
28
29
30 module StringMap = Map.Make(String)
31 module IntMap = Map.Make(struct type t = int let compare = compare end)
32
33 type typv_wrapper = Primitive of L.lltype | Complex of int
34
35
36 module FoldStack =
37    struct
38    type 'a t = { mutable c : 'a list; mutable len : int; }
39    exception Empty
40    let create () = { c = []; len = 0; }
41    let clear s = s.c <- []; s.len <- 0
42    let copy s = { c = s.c; len = s.len; }
43    let push x s = s.c <- x :: s.c; s.len <- s.len + 1
44    let pop s =
45      match s.c with
46      | hd::tl -> s.c <- tl; s.len <- s.len - 1; hd
```

```
    | []    -> raise Empty
  let top s =
    match s.c with
    | hd::_ -> hd
    | []    -> raise Empty
  let is_empty s = (s.c = [])
  let length s = s.len
  let iter f s = List.iter f s.c
  let fold f acc s = List.fold_left f acc s.c
  end ;;




let rec get_free_spec_list cntr =
   match cntr with
      0 -> []
    | _  -> (Free 0)::get_free_spec_list (cntr - 1)

let rec typify_as_int_access strs =
  match strs with
     x::xs -> (S.Id ("", S.Int, x))::typify_as_int_access xs
    |[] -> []

let rec add_with_identity l1 m =
   match l1 with
       x::xs -> if StringMap.mem x m then add_with_identity xs m
           else add_with_identity xs (StringMap.add x 1 m)
     | [] -> m

let diff l1 l2 = List.filter (fun x -> not (List.mem x l2)) l1

let rec add_with_count l1 m =
   match l1 with
       x::xs -> if StringMap.mem x m then add_with_count xs (StringMap.add x ((StringMap.find x
           m)+1) m)
           else add_with_count xs (StringMap.add x 1 m)
     | [] -> m

let remove_dups sl1 sl2 =
   let m = add_with_identity sl1 StringMap.empty in
   let m = add_with_identity sl2 m in

   List.map (fun (x,y) -> x) (StringMap.bindings (StringMap.filter (fun y x -> x == 1) m))

let remove_dups_eqone sl1 sl2 =
   let m = add_with_count sl1 StringMap.empty in
   let m = add_with_count sl2 m in

   List.map (fun (x,y) -> x) (StringMap.bindings (StringMap.filter (fun y x -> x == 1) m))

let rec extract_strings slist =
   match slist with
      [] -> []
    |x::xs -> (match x with S.Id (_, _, s) -> s::extract_strings xs | _ -> extract_strings xs)
```

```
102
103 let rec extract_tmult_info_helper e1 =
104     match e1 with
105         S.TensorId (_, _, _, sq) -> extract_strings sq
106       | S.TensorAssign (_, _, _, sq, _) -> extract_strings sq
107       | S.Binop (_, _, l, Ast.Mult, r) -> let (ls, rs) = extract_tmult_info l r in
108             remove_dups_eqone ls rs
109       | _ -> raise (Exceptions.CodegenException)
110 and
111 extract_tmult_info e1 e2 =
112     let left = extract_tmult_info_helper e1 in
113     let right = extract_tmult_info_helper e2 in
114
115     (left, right)
116
117
118 let rec make_slice_idx_copy_map vis sctr dctr map =
119     match vis with
120         Free _::xs -> let newmap = (IntMap.add sctr dctr map) in
121                 make_slice_idx_copy_map xs (sctr + 1) (dctr + 1) newmap
122       |Bound _::xs -> make_slice_idx_copy_map xs sctr (dctr+1) map
123       |[] -> map
124
125 let all_free xs =
126     List.for_all (fun x -> match x with Free _ -> true | _ -> false) xs
127
128 let all_bound xs =
129     List.for_all (fun x -> match x with Bound _ -> true | _ -> false) xs
130
131 let num_free xs =
132     List.length (List.filter (fun x -> match x with Free _ -> true | _ -> false) xs)
133
134 let strip_vis xs =
135     List.map (fun x -> match x with Bound y -> y | Free y -> y) xs
136
137 let get_free xs =
138     strip_vis (List.filter (fun x -> match x with Free _ -> true | _ -> false) xs)
139
140 let rec add_vis xs vis =
141     match (xs, vis) with
142         (x::xs, y::ys) ->
143             (match y with Bound _ -> (Bound x)::(add_vis xs ys) |
144               Free _ -> (Free x)::(add_vis xs ys))
145       | ([], []) -> []
146       | _ -> raise (Exceptions.CodegenException)
147
148 let get_first xs =
149     match xs with
150        x::xs -> x
151      | [] -> raise (COMPILE_ERROR "argument list empty")
152
153 let get_dim t =
154     match t with
155         S.Tensor x -> x
156       |_ -> raise (COMPILE_ERROR "Not a tensor?")
157
```

```
158 let rec get_vis_dummy_slist dim =
159     match dim with
160       0 -> []
161      |_ -> (S.Id ("", S.Int, "_DONOTUSE_WOW_"))::get_vis_dummy_slist (dim-1)
162
163 let translate program =
164   let context = L.global_context () in
165   let the_module = L.create_module context "LaTenS"
166   and i32_t = L.i32_type context
167   and i8_t  = L.i8_type  context
168   and i1_t  = L.i1_type  context
169   and i64_t = L.i64_type context
170   and void_t = L.void_type context
171   and f64_t = L.double_type context
172   and cstr_t = L.pointer_type (L.i8_type context)
173   and i64p_t = L.pointer_type (L.i64_type context)
174   and f64p_t = L.pointer_type (L.double_type context) in
175   (* We need special building types for the tensor stuff. but this
176      wrapper function currently gets us the primitive types we support
177      at this moment *)
178
179    let rec gen_pointer_map hdr acc uniq m cntr =
180       match acc with
181       [] -> m
182          |x::xs -> if not (StringMap.mem x uniq) then
183       gen_pointer_map hdr xs uniq m (cntr + 1)
184       else
185       gen_pointer_map hdr xs uniq ((L.build_add hdr (L.const_int i64_t (8 * (cntr + 1)))
              "hdrcpycalc")::m) (cntr + 1) in
186
187    let gen_pointer_map_wrap hdr1 hdr2 acc1 acc2 uniq =
188       let pmap1 = List.rev (gen_pointer_map hdr1 acc1 uniq [] 0) in
189       let pmap2 = List.rev (gen_pointer_map hdr2 acc2 uniq [] 0) in
190       pmap1@pmap2 in
191
192    let rec filter_sizes_by_pmap acc sizes uniq =
193       match (acc, sizes) with
194            ([], []) -> []
195          | (x::xs, y::ys) -> if StringMap.mem x uniq then
196              filter_sizes_by_pmap xs ys uniq
197       else
198          y::(filter_sizes_by_pmap xs ys uniq) in
199
200
201   let rec copy_header_static_internal src dst vis builder =
202      match vis with
203        [] -> ()
204      | Bound _::xs ->
205         let sintp1 = L.build_add src (L.const_int i64_t 8) "srcint" builder in
206         copy_header_static_internal sintp1 dst xs builder
207      | Free _::xs ->
208         let sptr = L.build_inttoptr src i64p_t "srcptr" builder in
209         let tptr = L.build_inttoptr dst i64p_t "dstptr" builder in
210         let sv = L.build_load sptr "memload" builder in
211         let ign = L.build_store sv tptr builder in
212         let sintp1 = L.build_add src (L.const_int i64_t 8) "srcint" builder in
```

```
213           let tintp1 = L.build_add dst (L.const_int i64_t 8) "dstint" builder in
214           copy_header_static_internal sintp1 tintp1 xs builder
215     in
216     let copy_header_static src dst vis builder =
217       let sint = (L.build_ptrtoint src i64_t "srcint" builder) in
218       let dint = (L.build_ptrtoint dst i64_t "dstint" builder) in
219       let sintp1 = (L.build_add sint (L.const_int i64_t 8) "sintp1" builder) in
220       let dintp1 = (L.build_add dint (L.const_int i64_t 8) "dintp1" builder) in
221       copy_header_static_internal sintp1 dintp1 vis builder
222     in
223     let ltype_of_typ = function
224       | S.Int -> Primitive i64_t
225       | S.Float -> Primitive f64_t
226       | S.String -> Primitive (L.pointer_type i8_t)
227       | S.Tensor dim -> Complex dim
228       | S.Void -> Primitive void_t
229     in
230     let ltype_of_typ_wrapper v =
231         let wrapped = ltype_of_typ v in
232         match wrapped with
233             Primitive x -> x
234           | Complex 0 -> f64_t
235           | Complex _ -> i64p_t
236     in
237     let build_tensor_header i name builder =
238         let pntr = L.build_array_malloc i64_t (L.const_int i64_t (i+2)) name builder in
239         ignore (L.build_store (L.const_int i64_t i) pntr builder);
240         pntr
241     in
242     let access_ith_true arr i builder sizeof =
243         let cast = L.build_ptrtoint arr i64_t "intcast" builder in
244         let offset = L.build_mul i sizeof "offsetcalc" builder in
245         let added = L.build_add cast offset "pmath" builder in
246         L.build_inttoptr added (L.pointer_type i64_t) "pointercast" builder
247     in
248     let access_ith arr i builder sizeof =
249         let iconst = L.const_int i64_t i in
250         let sof = L.const_int i64_t sizeof in
251         access_ith_true arr iconst builder sof
252     in
253
254     let rec assign_header hdr dims builder offset =
255         match dims with
256             | x::xs -> let access = access_ith hdr (offset+1) builder 8 in
257           ignore (L.build_store x access builder);
258           assign_header hdr xs builder (offset + 1)
259       | [] -> ()
260     in
261     let assign_header_static hdr dims builder offset =
262       let dimsconsts = List.map (fun x -> L.const_int i64_t x) dims in
263       assign_header hdr dimsconsts builder offset
264     in
265     let rec next_nobounds_internal vis builder =
266         match vis with
267             (Bound x):: xs -> (Bound x)::next_nobounds_internal xs builder
268       | (Free x)::xs ->
```

```
269       let tmpplace = L.build_load x "nextitload" builder in
270       let newval = L.build_add tmpplace (L.const_int i64_t 1) "iter_it" builder in
271       ignore (L.build_store newval tmpplace builder);
272        (Free x)::xs
273       | [] -> []
274     in
275     let next_nobounds vis builder =
276        next_nobounds_internal vis builder
277     in
278     (*
279     let rec static_memcpy_internal src dst cnt builder =
280         if cnt = 0 then () else
281             let sptr = L.build_inttoptr src i64p_t "srcptr" builder in
282             let tptr = L.build_inttoptr dst i64p_t "dstptr" builder in
283             let sv = L.build_load sptr "memload" builder in
284             ignore (L.build_store sv tptr "memstor" builder);
285             let sintp1 = L.build_add src (L.const_int i64_t 8) "srcint" builder in
286             let tintp1 = L.build_add dst (L.const_int i64_t 8) "dstint" builder in
287             static_memcpy_internal sintp1 tintp1 (cnt-1) builder
288     in
289     let static_memcpy src dst cnt builder =
290         static_memcpy_internal (L.build_ptrtoint src i64_t "srcint" builder)
291           (L.build_ptrtoint dst i64_t "dstint" builder) cnt builder
292     in *)
293
294     let rec static_memcpy_frompmap pmap dst cntr builder =
295         match pmap with
296           [] -> ()
297          |x::xs -> let sint = x builder in
298                    let tint = L.build_ptrtoint dst i64_t "dstint" builder in
299          let tintoffset = L.build_add tint (L.const_int i64_t (cntr*8)) "tintoffset" builder in
300          let sptr = L.build_inttoptr sint i64p_t "srcptr" builder in
301          let tptr = L.build_inttoptr tintoffset i64p_t "dstptr" builder in
302          let sv = L.build_load sptr "memload" builder in
303          ignore (L.build_store sv tptr builder);
304          static_memcpy_frompmap xs dst (cntr + 1) builder
305     in
306
307
308     (* Declare each global variable; remember its value in a map *)
309     let global_vars =
310       let global_var m vdecl =
311         let ltype = vdecl.S.vtyp in
312      let lname = vdecl.S.sname in
313      let lexpr = vdecl.S.vexpr in
314      let lval = (match ltype with
315          S.String ->
316            (match lexpr with
317         S.Sliteral (l,_,s) -> let ta = L.const_string context s in L.define_global lname ta
                the_module |
318         _ -> raise (TODO "Uhh buddy you gave me a non string here"))
319         | S.Int -> (match lexpr with
320            S.Iliteral (l,_,i) -> let ta = L.const_int i64_t i in L.define_global lname ta the_module |
321         _ -> raise (TODO "Expected int"))
322         | S.Float -> (match lexpr with
```

```
323           S.Fliteral (l,_,f) -> let ta = L.const_float f64_t f in L.define_global lname ta
                  the_module |
324       _ -> raise (TODO "Expected float"))
325       | S.Void -> raise (COMPILE_ERROR "I'm sorry, I can't let you do that Dave")
326       | S.Tensor _ -> raise (TODO "Dave, what have you been smoking?"))
327     in StringMap.add lname lval m in
328
329     (* let global_var m (t, n) =
330       let init = L.const_int (ltype_of_typv t) 0
331       in StringMap.add n (L.define_global n init the_module) m in *)
332     List.fold_left global_var StringMap.empty program.S.globals in
333
334   (* Declare printf(), which the print built-in function will call *)
335   let printf_t = L.var_arg_function_type i64_t [| L.pointer_type i8_t |] in
336   let printf_func = L.declare_function "printf" printf_t the_module in
337
338   let strlen_t = L.function_type i32_t [|L.pointer_type i8_t|] in
339   let strlen_func = L.declare_function "strlen" strlen_t the_module in
340
341   let sprintf_t = L.var_arg_function_type i64_t [| L.pointer_type i8_t; L.pointer_type i8_t |] in
342   let sprintf_func = L.declare_function "sprintf" sprintf_t the_module in
343
344   let atoi_t = L.function_type i64_t [| L.pointer_type i8_t|] in
345   let atoi_func = L.declare_function "atoi" atoi_t the_module in
346
347   let atof_t = L.function_type f64_t [| L.pointer_type i8_t|] in
348   let atof_func = L.declare_function "atof" atof_t the_module in
349
350   let fopen_t = L.function_type i64_t [| L.pointer_type i8_t; L.pointer_type i8_t|] in
351   let fopen_func = L.declare_function "fopen" fopen_t the_module in
352
353   let fclose_t = L.function_type i64_t [| i64_t|] in
354   let fclose_func = L.declare_function "fclose" fclose_t the_module in
355
356   let fread_t = L.function_type i32_t [| i64p_t; i32_t; i32_t; i64p_t|] in
357   let fread_func = L.declare_function "fread" fread_t the_module in
358
359   let fwrite_t = L.function_type i64_t [| i64p_t; i32_t; i32_t; i64p_t|] in
360   let fwrite_func = L.declare_function "fwrite" fwrite_t the_module in
361
362   let sqrt_t = L.function_type f64_t [| f64_t|] in
363   let sqrt_func = L.declare_function "sqrt" sqrt_t the_module in
364
365
366
367   (* Define each function (arguments and return type) so we can call it *)
368   let function_decls =
369     let function_decl m fdecl =
370       let name = fdecl.S.sfname
371       and formal_types =
372     Array.of_list (List.map (fun (t,_) -> ltype_of_typ_wrapper t) fdecl.S.sformals)
373       in let ftype = L.function_type (ltype_of_typ_wrapper fdecl.S.ftyp) formal_types in
374       StringMap.add name (L.define_function name ftype the_module, fdecl) m in
375     List.fold_left function_decl StringMap.empty program.S.functions in
376
377   (* Fill in the body of the given function *)
```

```
378   let build_function_body fdecl =
379     let the_scope = FoldStack.create () in FoldStack.push global_vars the_scope;
380     let (the_function, _) = StringMap.find fdecl.S.sfname function_decls in
381     let builder = L.builder_at_end context (L.entry_block the_function) in
382     let builder_ref = ref builder in
383
384     let int_format_str = L.build_global_stringptr "%d\n" "fmt" builder in
385     let flt_format_str = L.build_global_stringptr "%.6f\n" "fmt2" builder in
386     let fltcomma_format_str = L.build_global_stringptr "%.6f, " "fmt3" builder in
387     let concat_format_str = L.build_global_stringptr "%s%s" "concatfmt" builder in
388     let int_to_string_format_str = L.build_global_stringptr "%d" "itosfmt" builder in
389     let newline_str = L.build_global_stringptr "\n" "nlstr" builder in
390     let float_to_string_format_str = L.build_global_stringptr "%.6f" "ftosfmt" builder in
391
392
393     let rec gen_free_vars_from_names lst builder =
394         match lst with
395       [] -> ()
396      |x::xs -> let local = L.build_alloca i64_t x builder in
397       ignore (L.build_store (L.const_int i64_t 0) local);
398            let curr_stringmap = FoldStack.pop the_scope in
399       let strmap2 = StringMap.add x local curr_stringmap in
400       ignore (FoldStack.push strmap2 the_scope);
401       gen_free_vars_from_names xs builder
402     in
403
404     let rec gen_free_vars prefix cnt builder =
405         let nm = prefix ^ (string_of_int cnt) in
406         if cnt = 0 then [] else
407       let local = L.build_alloca i64_t nm builder in
408       ignore (L.build_store (L.const_int i64_t 0) local);
409       let curr_stringmap = FoldStack.pop the_scope in
410       let strmap2 = StringMap.add nm local curr_stringmap in
411       ignore (FoldStack.push strmap2 the_scope);
412       nm::(gen_free_vars prefix (cnt - 1)) builder in
413
414     (* Construct the function's "locals": formal arguments and locally
415        declared variables. Allocate each on the stack, initialize their
416        value, if appropriate, and remember their values in the "locals" map *)
417      let add_formal (t, n) p = L.set_value_name n p;
418     let local = L.build_alloca (ltype_of_typ_wrapper t) n builder in
419     ignore (L.build_store p local builder);
420     let curr_stringmap = FoldStack.pop the_scope in
421     let strmap2 = StringMap.add n local curr_stringmap in
422     FoldStack.push strmap2 the_scope in
423
424      List.iter2 add_formal fdecl.S.sformals
425        (Array.to_list (L.params the_function));
426     (* Return the value for a variable or formal argument *)
427     (* let lookup n = StringMap.find n.S.sname local_vars *)
428     let lookup_simple the_key already_found curr_map =
429       match already_found with
430       Some x -> Some x |
431       None -> if StringMap.mem the_key curr_map then Some (StringMap.find the_key curr_map)
432               else None
433     in
```

```
434     let lookup n ms =
435         let answer = FoldStack.fold (lookup_simple n) None ms in
436    match answer with
437        None -> raise (COMPILE_ERROR "query lookup on variable failed")
438      | Some x -> x
439     in
440     (* Construct code for an expression; return its value *)
441   let rec assign_tensor_floats pntr innerexprs offset builder =
442       match innerexprs with
443           [] -> ()
444           | x::xs -> let exprvalue = L.build_sitofp (expr builder x) f64_t "floattmp" builder in
445                let access = access_ith pntr offset builder 8 in
446                    let fpmem = L.build_malloc f64_t "factual" builder in
447                    let fpmemcast = L.build_ptrtoint fpmem i64_t "ptwtointforfpstorage" builder in
448            ignore (L.build_store exprvalue fpmem builder);
449                    ignore (L.build_store fpmemcast access builder);
450            assign_tensor_floats pntr xs (offset + 1) builder
451   and
452   get_vis slist builder =
453      (* TODO MAYBE FIX IF WE LET slicing work twice *)
454       match slist with
455           (S.Id (l, t, s))::xs -> (try (let e = expr builder (S.Id (l,t,s)) in
456                                 let sto = L.build_alloca i64_t "free_alloc" builder in
457               ignore (L.build_store e sto builder);
458               (Bound sto)::(get_vis xs builder)) with _ ->
459              (let sto = L.build_alloca i64_t "free_alloc" builder in
460           ignore (L.build_store (L.const_int i64_t 0) sto builder);
461           (Free sto)::(get_vis xs builder)))
462           | x::xs -> let e = expr builder x in
463                      (let sto = L.build_alloca i64_t "bnd_alloc" builder in
464                      ignore (L.build_store e sto builder);
465                       (Bound sto)::(get_vis xs builder))
466           |_ -> []
467   and
468
469   assign_tensor placetowrite i builder =
470       match i with
471          x::xs ->
472     (match x with
473           S.SExprDim(k, innerexprs) ->
474       let pntr = L.build_array_malloc i64_t
475            (L.const_int i64_t (List.length innerexprs)) "fstorage" builder in
476       ignore (assign_tensor_floats pntr innerexprs 0 builder);
477       let ptrtoint = (L.build_ptrtoint pntr i64_t "ptrtointfornlstorage" builder) in
478       ignore (L.build_store ptrtoint placetowrite builder);
479       let ptwint = (L.build_ptrtoint placetowrite i64_t "ptwtointfornlstorage" builder) in
480       let next = L.build_add ptwint (L.const_int i64_t 8) "nextptr" builder in
481       let nextptr = L.build_inttoptr next (L.pointer_type i64_t) "nextptrptr" builder in
482       ignore (assign_tensor nextptr xs builder);
483       ()
484    | S.STenDim(_,l) ->
485           let pntr = L.build_array_malloc i64_t
486                 (L.const_int i64_t (List.length l)) "fstarstorage" builder in
487           ignore (assign_tensor pntr l builder);
488       let ptrtoint = (L.build_ptrtoint pntr i64_t "ptrtointfornlstarstorage" builder) in
489       ignore (L.build_store ptrtoint placetowrite builder);
```

```
      let ptwint = (L.build_ptrtoint placetowrite i64_t "ptwtointfornlstorage" builder) in
      let next = L.build_add ptwint (L.const_int i64_t 8) "nextptr" builder in
      let nextptr = L.build_inttoptr next (L.pointer_type i64_t) "nextptrptr" builder in
      ignore (assign_tensor nextptr xs builder);
      ()
   )

   | [] -> ()
  and
  access_index_internal base exprlist builder =
      match exprlist with
          x::xs -> let ex = x in
         let sof = L.const_int i64_t 8 in
         let next = access_ith_true base ex builder sof in
              let deref = L.build_load next "deref" builder in
              let deref_pntr = L.build_inttoptr deref i64p_t "derefpntr" builder in
         access_index_internal deref_pntr xs builder
        (*
         | x::xs -> let ex = x in
         let sof = L.const_int i64_t 8 in
         let next = access_ith_true base ex builder sof in
         access_index_internal next xs builder *)
    |[]   -> base
  and
  access_index hdr exprs builder =
      let offset = L.build_load hdr "offsetload" builder in
      let offsetp1 = L.build_add offset (L.const_int i64_t 1) "plus1" builder in
      let sof = L.const_int i64_t 8 in
      let root = access_ith_true hdr offsetp1 builder sof in
      let rootload = L.build_load root "rootload" builder in
      access_index_internal rootload exprs builder
  and
  access_loc_internal base exprlist builder =
      match exprlist with
          x::xs -> let ex = x in
              let deref = L.build_load base "deref" builder in
              let sof = L.const_int i64_t 8 in
              let next = access_ith_true deref ex builder sof in
              let next_pntr = L.build_inttoptr next i64p_t "nextpntr" builder in
         access_loc_internal next_pntr xs builder
       | [] -> base
  and
  access_loc hdr exprs builder =
      let offset = L.build_load hdr "offsetload" builder in
      let offsetp1 = L.build_add offset (L.const_int i64_t 1) "plus1" builder in
      let sof = L.const_int i64_t 8 in
      let root = access_ith_true hdr offsetp1 builder sof in
      access_loc_internal root exprs builder
  and
  to_sexpr_from_strlist_force xs =
      List.map (fun x -> S.Sliteral ("TODO", S.String, x)) xs

  and
  rebound_internal clist builder carry the_function =
      match clist with
          (Free c,l)::xs ->
```

```
546         let l1load = L.build_load c "fiterload" builder in
547         let l2load = L.build_load l "siterload" builder in
548         let added = L.build_add l1load
549                 (L.build_bitcast carry i8_t "bc_fb" builder) "iteradd" builder in
550         (* Need to decrement to old value if overflow *)
551         let carrynew = L.build_icmp L.Icmp.Sge added l2load "itercmp" builder in
552             let merge_bb = L.append_block context "itermrg" the_function in
553             let then_bb = L.append_block context "iterthen" the_function in
554             let bldatthen = L.builder_at_end context then_bb in
555         ignore (L.build_store added c bldatthen);
556             ignore (L.build_br merge_bb bldatthen);
557             ignore (L.build_cond_br carrynew merge_bb then_bb builder);
558         rebound_internal xs (L.builder_at_end context merge_bb) carrynew the_function
559         | (Bound c, l)::xs -> rebound_internal xs builder carry the_function
560     | [] -> carry
561   and rebound vis ls builder =
562       let clist = List.combine vis ls in
563       rebound_internal clist builder (L.const_int i8_t 0)
564   and
565   (*
566   next_it ii builder the_function =
567       let curr = List.rev ii.current in
568       let limits = List.rev ii.limits in
569       ignore (next_nobounds ii.current builder);
570       let carryflag = rebound curr limits builder the_function in
571       ignore (L.build_cond_br carryflag ii.termination_branch ii.redo_branch builder)
572   and *)
573   malloc_callback base size offsets builder =
574       let loc = access_loc base offsets builder in
575       let arr = L.build_array_malloc i64_t size "mcarr" builder in
576       let arrint = L.build_ptrtoint arr i64_t "mcarrptrtoint" builder in
577       ignore (L.build_store arrint loc builder)
578   and
579   copy_callback sname soffname tname toffname builder =
580
581       let soffsets = typify_as_int_access soffname in
582       let toffsets = typify_as_int_access toffname in
583
584       let e = S.TensorId("", (S.Tensor 0), sname, soffsets) in
585       let e2 = S.TensorAssign("", (S.Tensor 0), tname, toffsets, e) in
586       ignore (expr builder e2); builder
587
588   and gen_names prefix cntr vals =
589       match vals with
590           [] -> []
591     |x::xs -> let stint = string_of_int cntr in
592         let name = (prefix ^ stint) in
593         let curr_stringmap = FoldStack.pop the_scope in
594         let strmap2 = StringMap.add name x curr_stringmap in
595         FoldStack.push strmap2 the_scope; name::(gen_names prefix (cntr + 1) xs)
596   and
597   gen_loads ptrs builder =
598       match ptrs with
599           x::xs -> (L.build_load (lookup x the_scope) "gl" builder)::gen_loads xs builder
600         | [] -> []
601   and
```

```
602   gen_loads_raw ptrs builder =
603       match ptrs with
604           x::xs -> (L.build_load x "glr" builder)::gen_loads_raw xs builder
605     | [] -> []
606   and
607   gen_sizes hdr cntr dim builder =
608       if cntr < dim then
609          let loc = access_ith hdr (cntr + 1) builder 8 in
610          (loc)::gen_sizes hdr (cntr + 1) dim builder
611       else []
612   and
613   get_last_in_list lst =
614       match lst with
615           x::y::rest -> get_last_in_list (y::rest)
616         | x::xs -> x
617         | [] -> raise (COMPILE_ERROR "Given empty list to get_last")
618   and
619   gen_cc_builderk sname dname names_access_src names_access_dst builder =
620       copy_callback sname names_access_src dname names_access_dst
621   and
622   gen_binop_callback tgt s1 s2 op names builder =
623       let laccess = S.TensorId("", S.Float, s1, (typify_as_int_access names)) in
624       let raccess = S.TensorId("", S.Float, s2, (typify_as_int_access names)) in
625       let bop = S.Binop ("", S.Float, laccess, op, raccess) in
626       let assign = S.TensorAssign("", S.Float, tgt, (typify_as_int_access names), bop) in
627       ignore (expr builder (assign));
628       builder
629   and
630
631   gen_tmult_binop_callback tgt s1 s2 tgtaccess_ s1access_ s2access_ builder =
632       let s1access = S.TensorId("", S.Float, s1, (typify_as_int_access s1access_)) in
633       let s2access = S.TensorId("", S.Float, s2, (typify_as_int_access s2access_)) in
634       let tgtaccess = S.TensorId("", S.Float, tgt, (typify_as_int_access tgtaccess_)) in
635       let bop = S.Binop("", S.Float, s1access, Ast.Mult, s2access) in
636       let bopadd = S.Binop("", S.Float, tgtaccess, Ast.Add, bop) in
637       let assign = S.TensorAssign("", S.Float, tgt, (typify_as_int_access tgtaccess_), bopadd) in
638
639       ignore (expr builder (assign));
640       builder
641   and
642   gen_tmult_callback_internal k free_var_names free_var_sizes builder =
643       match (free_var_names, free_var_sizes) with
644           ([], []) -> k builder
645     |(x::xs, y::ys) -> let cb = gen_tmult_callback_internal k xs ys in
646           let e1 = S.Assign ("", S.Int, x, S.Iliteral("",S.Int, 0)) in
647           let e2 = S.Binop ("", S.Int, S.Id("", S.Int, x), A.Less, S.Id("", S.Int, y)) in
648           let e3 = S.Assign ("", S.Int, x, S.Binop("", S.Int, S.Iliteral("", S.Int, 1), A.Add,
649               S.Id("", S.Int, x))) in
650           let body = S.Callback cb in
651           stmt builder (S.For (e1, e2, e3, body))
652
653   and
654   gen_tmult_callback k tgtname free_var_names free_var_sizes names builder =
655       let accumacc = S.TensorAssign("", S.Float, tgtname, (typify_as_int_access names), (S.Fliteral
            ("", S.Float, 0.0))) in
656       ignore (expr builder accumacc);
```

```
657         gen_tmult_callback_internal k free_var_names free_var_sizes builder
658     and
659     gen_print_float_llval llval builder =
660         ignore (L.build_call printf_func [| fltcomma_format_str; llval |] "printf_ignore" builder);
661         builder;
662     and
663     gen_printnewline_cb builder =
664         ignore (L.build_call printf_func [| newline_str |] "printf_ignore" builder);
665         builder;
666     and
667     gen_printf_ten tgt names names_done cntr builder =
668         match names with
669             x::xs ->
670           let loads_at_this_lvl = gen_loads (List.rev names_done) builder in
671           let sizes = gen_sizes tgt 0 (cntr + 1) builder in
672           let size = get_last_in_list sizes in
673           let sname_list = gen_names "_slicesize_" (List.length names_done) [size] in
674           let sname = get_first sname_list in
675          let kcall =
676               gen_printf_ten tgt (xs) (x::names_done) (cntr + 1) in
677               let e1 = S.Assign ("", S.Int, x, S.Iliteral("",S.Int, 0)) in
678               let e2 = S.Binop ("", S.Int, S.Id("", S.Int, x), A.Less, S.Id("", S.Int, sname)) in
679               let e3 = S.Assign ("", S.Int, x, S.Binop("", S.Int, S.Iliteral("", S.Int, 1), A.Add,
680                       S.Id("", S.Int, x))) in
681
682               let body = S.Callback kcall in
683               stmt builder (S.SBlock [S.For (e1,e2,e3,body); S.Callback gen_printnewline_cb])
684           | [] ->
685           let loads_at_this_lvl = gen_loads (List.rev names_done) builder in
686           let fptr = access_index tgt loads_at_this_lvl builder in
687           let fptrcast = L.build_bitcast fptr f64p_t "fpaccess" builder in
688               let fpval = L.build_load fptrcast "fpret" builder in
689               gen_print_float_llval fpval builder
690     and
691     gen_tensor_binop tgt k names names_done cntr builder =
692         match names with
693             x::xs ->
694           let loads_at_this_lvl = gen_loads (List.rev names_done) builder in
695           let sizes = gen_sizes tgt 0 (cntr + 1) builder in
696           let size = get_last_in_list sizes in
697           let mc = (malloc_callback tgt (L.build_load size "slicesizeload" builder)
698                   loads_at_this_lvl builder) in
699           let sname_list = gen_names "_slicesize_" (List.length names_done) [size] in
700           let sname = get_first sname_list in
701          let kcall =
702               gen_tensor_binop tgt k (xs) (x::names_done) (cntr + 1) in
703               let e1 = S.Assign ("", S.Int, x, S.Iliteral("",S.Int, 0)) in
704               let e2 = S.Binop ("", S.Int, S.Id("", S.Int, x), A.Less, S.Id("", S.Int, sname)) in
705               let e3 = S.Assign ("", S.Int, x, S.Binop("", S.Int, S.Iliteral("", S.Int, 1), A.Add,
706                       S.Id("", S.Int, x))) in
707               let body = S.Callback kcall in
708               stmt builder (S.For (e1,e2,e3,body))
709           | [] ->
710           let loads_at_this_lvl = gen_loads (List.rev names_done) builder in
711           let loc = access_loc tgt loads_at_this_lvl builder in
712           let ptr = L.build_malloc i64_t "abldrk" builder in
```

```
713           let ptrint = L.build_ptrtoint ptr i64_t "abldrkcast" builder in
714           ignore (L.build_store ptrint loc builder);
715               k (List.rev names_done) builder
716     and
717     gen_alloc_builderk names names_done base cntr builder =
718         match names with
719             x::xs ->
720           let loads_at_this_lvl = gen_loads (List.rev names_done) builder in
721           let sizes = gen_sizes base 0 (cntr + 1) builder in
722           let size = get_last_in_list sizes in
723           let mc = (malloc_callback base (L.build_load size "slicesizeload" builder)
724                 loads_at_this_lvl builder) in
725           let sname_list = gen_names "_slicesize_" (List.length names_done) [size] in
726           let sname = get_first sname_list in
727         let kcall =
728             gen_alloc_builderk (xs) (x::names_done) base (cntr + 1) in
729             let e1 = S.Assign ("", S.Int, x, S.Iliteral("",S.Int, 0)) in
730             let e2 = S.Binop ("", S.Int, S.Id("", S.Int, x), A.Less, S.Id("", S.Int, sname)) in
731             let e3 = S.Assign ("", S.Int, x, S.Binop("", S.Int, S.Iliteral("", S.Int, 1), A.Add,
732                     S.Id("", S.Int, x))) in
733             let body = S.Callback kcall in
734             stmt builder (S.For (e1,e2,e3,body))
735         | [] ->
736           let loads_at_this_lvl = gen_loads (List.rev names_done) builder in
737           let loc = access_loc base loads_at_this_lvl builder in
738           let ptr = L.build_malloc i64_t "abldrk" builder in
739           let ptrint = L.build_ptrtoint ptr i64_t "abldrkcast" builder in
740           ignore (L.build_store ptrint loc builder);
741           builder
742     and
743     gen_slice_builderk im names_todo names_done src dst k builder =
744         match names_todo with
745             x::xs ->
746
747           let loads_at_this_lvl = gen_loads (List.rev names_done) builder in
748
749           let cntr = IntMap.find (List.length names_done) im in
750           let sizes = gen_sizes src 0 (cntr + 1) builder in
751           let size = get_last_in_list sizes in
752           let mc = (malloc_callback dst (L.build_load size "slicesizeload" builder)
753                 loads_at_this_lvl builder) in
754           let sname_list = gen_names "_slicesize_" (List.length names_done) [size] in
755           let sname = get_first sname_list in
756         let kcall =
757             gen_slice_builderk im (xs) (x::names_done) src dst k in
758             let e1 = S.Assign ("", S.Int, x, S.Iliteral("",S.Int, 0)) in
759             let e2 = S.Binop ("", S.Int, S.Id("", S.Int, x), A.Less, S.Id("", S.Int, sname)) in
760             let e3 = S.Assign ("", S.Int, x, S.Binop("", S.Int, S.Iliteral("", S.Int, 1), A.Add,
761                     S.Id("", S.Int, x))) in
762             let body = S.Callback kcall in
763             stmt builder (S.For (e1,e2,e3,body))
764         | [] ->
765           let cc = k (List.rev names_done) builder in
766           stmt builder (S.Callback cc)
767     and
768
```

```
769   expr builder2 e =
770      builder_ref := builder2;
771      (expr2 e)
772   and
773   expr2 x=
774     match x with
775       | S.Noexpr _ -> L.const_int i64_t 0
776       | S.Id (line,t,s) -> L.build_load (lookup s the_scope) s !builder_ref
777       | S.Binop (line, t, e1, op, e2) ->
778          let e1' = expr !builder_ref e1
779          and e2' = expr !builder_ref e2 in
780              (match t with
781          S.Int ->
782            (match op with
783              A.Add    -> L.build_add e1' e2' "tmp" !builder_ref
784            | A.Sub    -> L.build_sub e1' e2' "tmp" !builder_ref
785            | A.Mult   -> L.build_mul e1' e2' "tmp" !builder_ref
786                | A.Div    -> L.build_sdiv e1' e2' "tmp" !builder_ref
787            | A.And    -> let b1 = (match L.type_of e1' with
788                                    i64_t -> L.build_icmp L.Icmp.Ne e1' (L.const_int i64_t 0)
                                                "booltmp" !builder_ref
789                                   |i1_t -> e1') in
790                                  let b2 = (match L.type_of e2' with
791                                    i64_t -> L.build_icmp L.Icmp.Ne e2' (L.const_int i64_t 0)
                                                "booltmp" !builder_ref
792                                   |i1_t -> e2') in
793                                  L.build_and b1 b2 "tmp" !builder_ref
794            | A.Or     -> let b1 = (match L.type_of e1' with
795                                    i64_t -> L.build_icmp L.Icmp.Ne e1' (L.const_int i64_t 0)
                                                "booltmp" !builder_ref
796                                   |i1_t -> e1') in
797                                  let b2 = (match L.type_of e2' with
798                                    i64_t -> L.build_icmp L.Icmp.Ne e2' (L.const_int i64_t 0)
                                                "booltmp" !builder_ref
799                                   |i1_t -> e2') in
800                                  L.build_or b1 b2 "tmp" !builder_ref
801            | A.Equal  -> L.build_icmp L.Icmp.Eq e1' e2' "tmp" !builder_ref
802            | A.Neq    -> L.build_icmp L.Icmp.Ne e1' e2' "tmp" !builder_ref
803            | A.Less   -> L.build_fcmp L.Fcmp.Olt (L.build_sitofp e1' f64_t "ftmp" !builder_ref)
                  (L.build_sitofp e2' f64_t "ftmp" !builder_ref) "tmp" !builder_ref
804            | A.Leq    -> L.build_fcmp L.Fcmp.Ole (L.build_sitofp e1' f64_t "ftmp" !builder_ref)
                  (L.build_sitofp e2' f64_t "ftmp" !builder_ref) "tmp" !builder_ref
805            | A.Greater -> L.build_fcmp L.Fcmp.Ogt (L.build_sitofp e1' f64_t "ftmp" !builder_ref)
                  (L.build_sitofp e2' f64_t "ftmp" !builder_ref) "tmp" !builder_ref
806            | A.Geq    -> L.build_fcmp L.Fcmp.Oge (L.build_sitofp e1' f64_t "ftmp" !builder_ref)
                  (L.build_sitofp e2' f64_t "ftmp" !builder_ref) "tmp" !builder_ref
807            )
808           | S.Float ->
809         let e1f = L.build_fpcast (L.build_sitofp e1' f64_t "cast_tmp" !builder_ref) f64_t
              "fpcast_eo" !builder_ref in
810         let e2f = L.build_fpcast (L.build_sitofp e2' f64_t "cast2_tmp" !builder_ref) f64_t
              "fpcast_eo" !builder_ref in
811         (match op with
812               A.Add    -> L.build_fadd
813             | A.Sub    -> L.build_fsub
814             | A.Mult   -> L.build_fmul
```

```
            | A.Div    -> L.build_fdiv
        | _        -> raise(Failure("This operation shouldn't return a float, something is wrong"))
          )
      e1f e2f "tmp" !builder_ref
         | S.String ->
       (match op with
        A.Concat ->
        let l1 = L.build_call strlen_func [|e1'|] "strlen1" !builder_ref in
        let l2 = L.build_call strlen_func [|e2'|] "strlen2" !builder_ref in
        let sum = (L.build_add (L.build_add l1 l2 "add_tmp" !builder_ref) (L.const_int i32_t 1) "add_tmp" !builder_ref) in
        let mem = L.build_array_malloc i8_t sum "malloc_tmp" !builder_ref in
        ignore(L.build_call sprintf_func [| mem; concat_format_str; e1'; e2' |] "sprintf_tmp" !builder_ref);
        mem
      |  _ -> raise(Failure("Can't use this operation on strings, something is wrong"))
      )
             | S.Tensor dim ->
       (match op with Ast.Mult -> let (luse, ruse) =
        extract_tmult_info e1 e2 in
        let tuse = remove_dups_eqone luse ruse in
        let tmap = List.fold_left (fun m e -> StringMap.add e () m) StringMap.empty tuse in
                 let s1dim = get_dim (Sm.extract_type e1) in
        let sizesformult = gen_sizes (e1') 0 s1dim !builder_ref in
        let sizesfiltered = filter_sizes_by_pmap luse sizesformult tmap in

        let thdr = (
        if List.length tuse > 0 then
        let thdr = build_tensor_header dim "static_ten" !builder_ref in
        let shdr1int = L.build_ptrtoint e1' i64_t "hdrint" !builder_ref in
        let shdr2int = L.build_ptrtoint e2' i64_t "hdrint2" !builder_ref in



        let pmap = gen_pointer_map_wrap shdr1int shdr2int luse ruse tmap in
        static_memcpy_frompmap pmap thdr 1 !builder_ref;
        thdr
        else
        (L.const_float f64_t 0.0)) in


       let hdraddr = L.build_alloca (L.type_of thdr) "tgt_header" !builder_ref in
       let s1addr = L.build_alloca (L.type_of e1') "s1_header" !builder_ref in
       let s2addr = L.build_alloca (L.type_of e2') "s2_header" !builder_ref in
       ignore (L.build_store thdr hdraddr !builder_ref);
       ignore (L.build_store e1' s1addr !builder_ref);
       ignore (L.build_store e2' s2addr !builder_ref);

       let sm = StringMap.empty in
       let sm2 = StringMap.add "_tgt_header_" hdraddr sm in
       let sm3 = StringMap.add "_s1_header_" s1addr sm2 in
       let sm4 = StringMap.add "_s2_header_" s2addr sm3 in


       FoldStack.push sm4 the_scope;
```

```
868         gen_free_vars_from_names tuse !builder_ref;
869         gen_free_vars_from_names (diff luse tuse) !builder_ref;
870         let sizenames = gen_names "_tmult_size_inner_" 0 sizesfiltered in
871         let lcb = gen_tmult_binop_callback "_tgt_header_" "_s1_header_" "_s2_header_" tuse luse
                ruse in
872         let cb = gen_tmult_callback lcb "_tgt_header_" (diff luse tuse) sizenames in
873
874
875         let newbuilder = (
876         if List.length tuse > 0 then
877             gen_tensor_binop thdr cb tuse [] 0 !builder_ref
878         else
879             cb [] !builder_ref) in
880
881         builder_ref := newbuilder;
882         FoldStack.pop the_scope;
883
884         L.build_load hdraddr "hdraddr_load" newbuilder
885         | _ ->
886         let hdr = build_tensor_header dim "static_ten" !builder_ref in
887         let frees = get_free_spec_list (dim) in
888         copy_header_static e1' hdr frees !builder_ref;
889         let dummy_vi = get_vis_dummy_slist dim in
890         let vis = get_vis dummy_vi !builder_ref in
891
892         let hdraddr = L.build_alloca (L.type_of hdr) "tgt_header" !builder_ref in
893         let s1addr = L.build_alloca (L.type_of e1') "s1_header" !builder_ref in
894         let s2addr = L.build_alloca (L.type_of e2') "s2_header" !builder_ref in
895         ignore (L.build_store hdr hdraddr !builder_ref);
896         ignore (L.build_store e1' s1addr !builder_ref);
897         ignore (L.build_store e2' s2addr !builder_ref);
898
899         let sm = StringMap.empty in
900         let sm2 = StringMap.add "_tgt_header_" hdraddr sm in
901         let sm3 = StringMap.add "_s1_header_" s1addr sm2 in
902         let sm4 = StringMap.add "_s2_header_" s2addr sm3 in
903
904         FoldStack.push (sm4) the_scope;
905         let names = gen_names "_slicenot_" 0 (strip_vis vis) in
906         let k = gen_binop_callback "_tgt_header_" "_s1_header_" "_s2_header_" op in
907         let newbuilder = gen_tensor_binop hdr k names [] 0 !builder_ref in
908         builder_ref := newbuilder;
909
910         ignore (FoldStack.pop the_scope);
911                 hdr)
912       )
913       | S.Unop(line, t, op, e) ->
914          let e' = expr !builder_ref e in
915            (match op with
916             A.Neg    -> if t = S.Int then L.build_neg e' "tmp" !builder_ref else L.build_fsub
                    (L.const_float f64_t 0.0) e' "tmp" !builder_ref
917           | A.Not    -> L.build_not (L.build_icmp L.Icmp.Ne e' (L.const_int i64_t 0) "nottmp"
                   !builder_ref) "tmp" !builder_ref)
918       | S.Assign (line, t, s, e) -> let e' = if t = S.Float then (L.build_sitofp (expr !builder_ref
            e) f64_t "cast_tmp" !builder_ref) else expr !builder_ref e in
919                   ignore (L.build_store e' (lookup s the_scope) !builder_ref); e'
```

```
920       (* TODO do calls *)
921       | S.Call (line, t, fd, sexprlist) ->
922          (if (compare fd.S.sfname "print") = 0 then
923              let fet = (get_first sexprlist) in
924              let ftype = Sm.extract_type fet in
925              match ftype with
926                S.Int ->
927                   L.build_call printf_func [| int_format_str ; (expr !builder_ref (get_first
                           sexprlist)) |] "printf_ignore" !builder_ref
928              | S.Float ->
929                   L.build_call printf_func [| flt_format_str ; (expr !builder_ref fet) |]
                           "printf_ignore" !builder_ref
930              | S.String ->
931                   L.build_call printf_func [| (expr !builder_ref fet) |] "printf_ignore" !builder_ref
932              | S.Tensor dim ->
933                    let e' = expr !builder_ref (get_first sexprlist) in
934                    FoldStack.push StringMap.empty the_scope;
935                    let names = gen_free_vars "_printf_prefix_" dim !builder_ref in
936                    let newbuilder = gen_printf_ten e' names [] 0 !builder_ref in
937                    builder_ref := newbuilder;
938                    ignore (FoldStack.pop the_scope);
939                    L.const_int i64_t 0
940           else if (compare fd.S.sfname "dim") = 0 then
941         let ptr = access_ith_true (expr !builder_ref (get_first sexprlist)) (L.build_add (expr
                !builder_ref (List.nth sexprlist 1)) (L.const_int i64_t 1) "addtmp" !builder_ref)
                !builder_ref (L.const_int i32_t 8) in
942         L.build_load ptr "tenresult" !builder_ref
943       else if (compare fd.S.sfname "ftoi") = 0 then
944         L.build_fptosi (expr !builder_ref (get_first sexprlist)) i64_t "inttmp" !builder_ref
945
946       else if (compare fd.S.sfname "itof") = 0 then
947         L.build_sitofp (expr !builder_ref (get_first sexprlist)) f64_t "floattmp" !builder_ref
948
949       else if (compare fd.S.sfname "ftos") = 0 then
950                let mem = L.build_array_malloc i8_t (L.const_int i32_t 50) "malloc_tmp" !builder_ref
                       in
951            ignore(L.build_call sprintf_func [|mem; float_to_string_format_str; (expr !builder_ref
                   (get_first sexprlist))|] "sprintf_tmp" !builder_ref);
952            mem
953
954       else if (compare fd.S.sfname "itos") = 0 then
955                let mem = L.build_array_malloc i8_t (L.const_int i32_t 50) "malloc_tmp" !builder_ref
                       in
956            ignore(L.build_call sprintf_func [|mem; int_to_string_format_str; (expr !builder_ref
                   (get_first sexprlist))|] "sprintf_tmp" !builder_ref);
957            mem
958
959       else if (compare fd.S.sfname "stoi") = 0 then
960            L.build_call atoi_func [|expr !builder_ref (get_first sexprlist)|] "atoi_tmp" !builder_ref
961
962       else if (compare fd.S.sfname "stof") = 0 then
963            L.build_call atof_func [|expr !builder_ref (get_first sexprlist)|] "atof_tmp" !builder_ref
964
965       else if (compare fd.S.sfname "fopen") = 0 then
966            L.build_call fopen_func [|(expr !builder_ref (get_first sexprlist)); (expr !builder_ref
                   (List.nth sexprlist 1))|] "fopen_tmp" !builder_ref
```

```
967
968     else if (compare fd.S.sfname "fclose") = 0 then
969          L.build_call fclose_func [|(expr !builder_ref (get_first sexprlist))|] "fclose_tmp"
                 !builder_ref
970
971     else if (compare fd.S.sfname "fread") = 0 then
972          let size = L.build_trunc (expr !builder_ref (get_first sexprlist)) i32_t "trunctmp"
                 !builder_ref in
973          let ptr = L.build_array_malloc i8_t size "malloc_tmp" !builder_ref in
974          let castptr = L.build_bitcast ptr i64p_t "bitcast_tmp" !builder_ref in
975              let nmemb = L.const_int i32_t 1 in
976          let file = L.build_inttoptr (expr !builder_ref (List.nth sexprlist 1)) i64p_t "cast_tmp"
                 !builder_ref in
977          ignore(L.build_call fread_func [|castptr; size; nmemb; file|] "fread_tmp" !builder_ref);
978          ptr
979
980
981     else if (compare fd.S.sfname "fwrite") = 0 then
982          let ptr = expr !builder_ref (get_first sexprlist) in
983          let size = L.build_call strlen_func [|ptr|] "strlen" !builder_ref in
984          let nmemb = L.const_int i32_t 1 in
985          let file = L.build_inttoptr (expr !builder_ref (List.nth sexprlist 1)) i64p_t "cast_tmp"
                 !builder_ref in
986          L.build_call fwrite_func [|(L.build_bitcast ptr i64p_t "bitcast_tmp" !builder_ref); size;
                 nmemb; file|] "fwrite_tmp" !builder_ref
987
988
989
990    else if (compare fd.S.sfname "sqrt") = 0 then
991        L.build_call sqrt_func [|(expr !builder_ref (get_first sexprlist))|] "sqrt_tmp" !builder_ref
992
993
994    else if (compare fd.S.sfname "strlen") = 0 then
995        L.build_call strlen_func [|expr !builder_ref (get_first sexprlist)|] "strlen" !builder_ref
996
997     else
998          let (fdef, fdecl) = StringMap.find fd.S.sfname function_decls in
999     let actuals = List.rev (List.map (expr !builder_ref) (List.rev sexprlist)) in
1000    let result = (match fdecl.S.ftyp with S.Void -> ""
1001                                        | _ -> fd.S.sfname ^ "_result") in
1002         L.build_call fdef (Array.of_list actuals) result !builder_ref
1003         )
1004
1005      | S.Fliteral (line,_,f) -> L.const_float f64_t f
1006      | S.Iliteral (line,_,i) -> L.const_int i64_t i
1007      | S.Sliteral (line,_,s) -> L.build_global_stringptr s s !builder_ref
1008      | S.Itensor (line,_,S.STenDim(l,i)) ->
1009          let hdr = build_tensor_header (List.length l) "immediate_tmp" !builder_ref in
1010     ignore (assign_header_static hdr l !builder_ref 0);
1011
1012     let datptroffset = access_ith hdr (1 + (List.length l)) !builder_ref 8 in
1013
1014     ignore (assign_tensor datptroffset [S.STenDim(l,i)] !builder_ref);
1015     hdr
1016      | S.Itensor (line,_,S.SExprDim(i, l)) ->
1017          let hdr = build_tensor_header 1 "immediate_tmp" !builder_ref in
```

```
1018        ignore (assign_header_static hdr [List.length l] !builder_ref 0);
1019        let datptroffset = access_ith hdr 2 !builder_ref 8 in
1020        ignore (assign_tensor datptroffset [S.SExprDim(i,l)] !builder_ref);
1021        hdr
1022         | S.BindExpr (line, t, S.InferredBind (inner_line, n, e)) -> let alloc = L.build_alloca
                 (ltype_of_typ_wrapper t) n !builder_ref in
1023            let e' = expr !builder_ref e in
1024        let curr_stringmap = FoldStack.pop the_scope in
1025        ignore (L.build_store e' alloc !builder_ref);
1026        let strmap2 = StringMap.add n alloc curr_stringmap in
1027        FoldStack.push strmap2 the_scope; e';
1028         | S.BindExpr (line, t, S.TensorBind (inner_line, s, qllist)) ->
1029            let hdr = build_tensor_header (List.length qllist) "dynamic_ten" !builder_ref in
1030        let qvals = List.map (fun x -> expr !builder_ref x) qllist in
1031
1032            FoldStack.push StringMap.empty the_scope;
1033        ignore (assign_header hdr qvals !builder_ref 0);
1034        let inames = gen_free_vars "_free_vars" (List.length qvals) !builder_ref in
1035            let newbuilder = gen_alloc_builderk inames [] hdr 0 !builder_ref in
1036        builder_ref := newbuilder;
1037            FoldStack.pop the_scope;
1038        let alloc = L.build_alloca (L.pointer_type i64_t) s !builder_ref in
1039        ignore (L.build_store hdr alloc !builder_ref);
1040            let curr_stringmap = FoldStack.pop the_scope in
1041        let strmap2 = StringMap.add s alloc curr_stringmap in
1042        FoldStack.push strmap2 the_scope; hdr
1043         | S.BindExpr (line, t, S.TensorBindAndAssign(inner_line, s,q,e)) ->
1044            ignore (expr !builder_ref (S.BindExpr(line, t, S.TensorBind(inner_line, s, q))));
1045        expr !builder_ref (S.TensorAssign(inner_line, t, s, q, e))
1046
1047         | S.TensorAssign (line, t, s, slist, e) ->
1048            if List.length slist = 0 then
1049           expr !builder_ref (S.Assign ("", S.Float, s, e))
1050        else
1051            let e' = expr !builder_ref e in
1052            let vis = get_vis slist !builder_ref in
1053            if all_bound vis then
1054           if (t = S.Float || t = S.Int) then
1055             let tptr = L.build_load (lookup s the_scope) s !builder_ref in
1056             let lds = (gen_loads_raw (strip_vis vis) !builder_ref) in
1057             let fptr = access_index tptr lds !builder_ref in
1058             let fptrcast = L.build_bitcast fptr f64p_t "fpaccess" !builder_ref in
1059             ignore (L.build_store (L.build_sitofp (e') f64_t "floatcast" !builder_ref) fptrcast
                     !builder_ref); e'
1060           else
1061             let tptr = L.build_load (lookup s the_scope) s !builder_ref in
1062             let lds = (gen_loads_raw (strip_vis vis) !builder_ref) in
1063             let iptr = access_loc tptr lds !builder_ref in
1064             ignore (L.build_store e' iptr !builder_ref); e'
1065
1066            else
1067               (ignore (L.build_store e' (lookup s the_scope) !builder_ref);
1068               e')
1069         | S.TensorId (line, t, s, slist) ->
1070            if List.length slist = 0 then
1071           expr !builder_ref (S.Id ("", S.Float, s))
```

```
1072      else
1073          let vis = get_vis slist !builder_ref in
1074          if all_bound vis then
1075              let tptr = L.build_load (lookup s the_scope) s !builder_ref in
1076          let lds = (gen_loads_raw (strip_vis vis) !builder_ref) in
1077              let fptr = access_index tptr lds !builder_ref in
1078          if t = S.Float then
1079                  let fptrcast = L.build_bitcast fptr f64p_t "fpaccess" !builder_ref in
1080                  L.build_load fptrcast "fpret" !builder_ref
1081              else
1082             L.build_ptrtoint fptr i64_t "icast" !builder_ref
1083          else if all_free vis then
1084              L.build_load (lookup s the_scope) s !builder_ref
1085          else
1086              let nf = num_free vis in
1087              let shdr = L.build_load (lookup s the_scope) s !builder_ref in
1088              let hdr = build_tensor_header nf "cpy_hdr" !builder_ref in
1089          let hdraddr = L.build_alloca (L.type_of hdr) "tgt_header" !builder_ref in
1090          ignore (L.build_store hdr hdraddr !builder_ref);
1091              copy_header_static shdr hdr vis !builder_ref;
1092
1093          FoldStack.push (StringMap.add "_tgt_header" hdraddr StringMap.empty) the_scope;
1094
1095          let names = gen_names "_slicenot_" 0 (strip_vis vis) in
1096              let names_spec = add_vis names vis in
1097              let names_free = get_free names_spec in
1098          let im = make_slice_idx_copy_map vis 0 0 IntMap.empty in
1099          let k = gen_cc_builderk s "_tgt_header" names in
1100              let newbuilder = gen_slice_builderk im names_free [] shdr hdr k !builder_ref in
1101          FoldStack.pop the_scope;
1102          builder_ref := newbuilder;
1103
1104              hdr
1105     (* Invoke "f builder" if the current block doesn't already
1106        have a terminal (e.g., a branch). *)
1107     and add_terminal builder f =
1108       match L.block_terminator (L.insertion_block builder) with
1109    Some _ -> ()
1110      | None -> ignore (f builder)
1111
1112     and stmt builder = function
1113    S.SBlock sl -> FoldStack.push StringMap.empty the_scope; let bld = List.fold_left stmt builder
           sl in
1114       ignore (FoldStack.pop the_scope); bld
1115       | S.Expr e -> builder_ref := builder; ignore (expr builder e); !builder_ref
1116       | S.Return e -> ignore (match fdecl.S.ftyp with
1117      S.Void -> L.build_ret_void builder
1118    | _ ->
1119    L.build_ret (expr builder e) builder); builder
1120       | S.Callback f -> f builder
1121       | S.If (predicate, then_stmt, else_stmt) ->
1122             let p = expr builder predicate in
1123             let bool_val = (match L.type_of p with
1124                  i64_t -> L.build_icmp L.Icmp.Ne p (L.const_int i64_t 0) "booltmp" !builder_ref
1125                 |i1_t -> p) in
1126     let merge_bb = L.append_block context "merge" the_function in
```

```
    let then_bb = L.append_block context "then" the_function in
    add_terminal (stmt (L.builder_at_end context then_bb) then_stmt)
      (L.build_br merge_bb);

    let else_bb = L.append_block context "else" the_function in
    add_terminal (stmt (L.builder_at_end context else_bb) else_stmt)
      (L.build_br merge_bb);

    ignore (L.build_cond_br bool_val then_bb else_bb builder);
    L.builder_at_end context merge_bb

      | S.While (predicate, body) ->
     let pred_bb = L.append_block context "while" the_function in
     ignore (L.build_br pred_bb builder);

     let body_bb = L.append_block context "while_body" the_function in
     add_terminal (stmt (L.builder_at_end context body_bb) body)
       (L.build_br pred_bb);

     let pred_builder = L.builder_at_end context pred_bb in
     let p = expr pred_builder predicate in
         let bool_val = (match L.type_of p with
                i64_t -> L.build_icmp L.Icmp.Ne p (L.const_int i64_t 0) "booltmp" !builder_ref
               |i1_t -> p) in

     let merge_bb = L.append_block context "merge" the_function in
     ignore (L.build_cond_br bool_val body_bb merge_bb pred_builder);
     L.builder_at_end context merge_bb

      | S.For (e1, e2, e3, body) ->
            stmt builder (S.SBlock [S.Expr e1; S.While (e2, S.SBlock [body; S.Expr e3])])
    in

    (* Build the code for each statement in the function *)
    let builder = stmt builder fdecl.S.sbody in

    (* Add a return if the last block falls off the end *)
    add_terminal builder (match fdecl.S.ftyp with
        S.Void -> L.build_ret_void
      | t -> L.build_ret (L.const_int (ltype_of_typ_wrapper t) 0))
  in

  List.iter build_function_body program.S.functions;
  the_module
```

9.8 parser.mly

```
%{ open Ast
   open Lexing

let parse_error msg =
  raise ( Exceptions.ParserException( ((rhs_start_pos 1).pos_lnum, ((rhs_start_pos 1).pos_cnum
      -(rhs_start_pos 1).pos_bol), (rhs_end_pos 1).pos_lnum, ((rhs_end_pos 1).pos_cnum -
```

```
      (rhs_end_pos 1).pos_bol), msg)))
%}


%token SEMI LPAREN RPAREN LBRACE RBRACE COMMA
%token PLUS MINUS TIMES DIVIDE ASSIGN NOT LET
%token LBRACKET RBRACKET UNDER CONCAT
%token EQ NEQ LT LEQ GT GEQ AND OR
%token RETURN IF ELSE FOR WHILE
%token FLOAT STRING VOID INT

%token <string * int> ILITERAL
%token <string * float> FLITERAL
%token <string * string> SLITERAL
%token <string * string> ID
%token <string * string> COMMENT
%token EOF


%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left OR
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS CONCAT
%left TIMES DIVIDE
%right NOT NEG


%start program
%type <Ast.program> program

%%

program: decls EOF { ((List.rev (fst $1)), (List.rev (snd $1))) }

decls: /* nothing */ { [], [] }
  | decls vdecl SEMI    { ($2 :: fst $1), snd $1 }
  | decls fdecl    { fst $1, ($2 :: snd $1) }

fdecl: comment typ ID LPAREN formals_opt RPAREN comment LBRACE stmt_list RBRACE
   { { comm = List.rev($1); typ = $2; fname = snd $3; formals = List.rev($5);
         body = Block($7, List.rev($9)) } }

comment: /* nothing */ { [] }
  | comment COMMENT { $2 :: $1 }

formals_opt: /* nothing */ { [] }
  | var_typ ID              { [($1,snd $2)] }
  | formals_opt COMMA var_typ ID { ($3,snd $4) :: $1 }

typ: FLOAT { Float }
   | STRING {String}
   | VOID { Void }
```

```
   | INT { Int }
   | ID UNDER LBRACE qual_list_expr { TensorType($4) }

var_typ: FLOAT { Float }
  | STRING { String }
  | INT { Int }
  | ID UNDER LBRACE qual_list_expr { TensorType($4) }

vdecl: LET ID UNDER LBRACE qual_list_expr { TensorBind(fst $2, snd $2, $5)}
   | LET ID UNDER LBRACE qual_list_expr ASSIGN expr { TensorBindAndAssign(fst $2, snd $2, $5, $7) }
   | LET ID ASSIGN expr { InferredBind(fst $2, snd $2, $4) }

stmt_list: /* nothing */ {[] }
  | stmt_list stmt { $2 :: $1 }

stmt: comment expr SEMI               { Expr (List.rev($1), $2) }
  | comment RETURN SEMI               { Return (List.rev($1), Noexpr) }
  | comment RETURN expr SEMI                        { Return(List.rev($1), $3) }
  | comment LBRACE stmt_list RBRACE      { Block(List.rev($1), List.rev($3)) }
  | comment IF LPAREN expr RPAREN stmt %prec NOELSE    { If(List.rev($1), $4, $6, Block([], [])) }
  | comment IF LPAREN expr RPAREN stmt ELSE stmt      { If(List.rev($1), $4, $6, $8) }
  | comment FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN stmt  { For(List.rev($1), $4, $6,
      $8, $10) }
  | comment WHILE LPAREN expr RPAREN stmt          { While(List.rev($1), $4, $6) }


qual_list_expr: expr RBRACE   { [$1] }
  | expr COMMA qual_list_expr    { $1::$3 }


ten_list: expr COMMA ten_list { $1::$3 }
  | expr                    { [$1] }


expr:
    ILITERAL        { Iliteral(fst $1, snd $1) }
  | SLITERAL                  { Sliteral(fst $1, snd $1) }
  | FLITERAL        { Fliteral(fst $1, snd $1) }
  | expr PLUS expr       { Binop(get_line_expr $1, $1, Add, $3) }
  | expr MINUS expr         { Binop(get_line_expr $1, $1, Sub, $3) }
  | expr TIMES expr         { Binop(get_line_expr $1, $1, Mult, $3) }
  | expr DIVIDE expr        { Binop(get_line_expr $1, $1, Div, $3) }
  | expr CONCAT expr   { Binop(get_line_expr $1, $1, Concat, $3) }
  | expr EQ expr              { Binop(get_line_expr $1, $1, Equal, $3) }
  | expr NEQ expr           { Binop(get_line_expr $1, $1, Neq, $3) }
  | expr LT expr              { Binop(get_line_expr $1, $1, Less, $3) }
  | expr LEQ expr           { Binop(get_line_expr $1, $1, Leq, $3) }
  | expr GT expr              { Binop(get_line_expr $1, $1, Greater, $3) }
  | expr GEQ expr           { Binop(get_line_expr $1, $1, Geq, $3) }
  | expr AND expr           { Binop(get_line_expr $1, $1, And, $3) }
  | expr OR expr              { Binop(get_line_expr $1, $1, Or, $3) }
  | MINUS expr %prec NEG  { Unop(get_line_expr $2, Neg, $2) }
  | PLUS expr       { $2 (* Unary plus is redundant; ignore it *) }
  | NOT expr                  { Unop(get_line_expr $2, Not, $2) }
  | ID             { Id(fst $1, snd $1) }
  | vdecl  { BindExpr(get_line_bind $1, $1) }
```

```
  | ID ASSIGN expr          { Assign(fst $1, snd $1, $3) }
  | LPAREN expr RPAREN  { $2 }
  | ID LPAREN actuals_opt RPAREN { Call(fst $1, snd $1, $3) }
  | ID UNDER LBRACE qual_list_expr { TensorId(fst $1, snd $1, $4) }
  | ID UNDER LBRACE qual_list_expr ASSIGN expr { TensorAssign(fst $1, snd $1, $4, $6) }
  | LBRACKET ten_list RBRACKET { Tensor(get_line_expr (List.hd $2), $2) }


expr_opt:
   /* nothing */ { Noexpr }
  | expr { $1 }

actuals_opt:
   /* nothing */ { [] }
  | actuals_list { List.rev $1 }

actuals_list:
   expr { [$1] }
| actuals_list COMMA expr { $3 :: $1 }
```

9.9 sast.ml

```
1  (*open Ast*)
2
3
4  type t =
5    Int
6  | Float
7  | String
8  | Void
9  | Tensor of int
10
11 let string_of_styp = function
12     Int -> "int"
13   | Float -> "float"
14   | String -> "string"
15   | Void -> "void"
16   | Tensor(i) -> (string_of_int i) ^ "-tensor"
17
18 type sbind = InferredBind of Ast.line * string * sexpr
19 | TensorBind of Ast.line * string * squalifier
20 | TensorBindAndAssign of Ast.line * string * squalifier * sexpr
21 and
22 sexpr =
23   Id of Ast.line * t * string
24 | Noexpr of t
25 | Binop of Ast.line * t * sexpr * Ast.op * sexpr
26 | Unop of Ast.line * t * Ast.uop * sexpr
27 | Assign of Ast.line * t * string * sexpr
28 | Call of Ast.line * t * sfunc_decl * sexpr list
29 | Fliteral of Ast.line * t * float
30 | Iliteral of Ast.line * t * int
31 | Sliteral of Ast.line * t * string
32 | TensorId of Ast.line * t * string * squalifier
```

```
| TensorAssign of Ast.line * t * string * squalifier * sexpr
| Itensor of Ast.line * t * stimmediate
| BindExpr of Ast.line * t * sbind
and squalifier = sexpr list

and stimmediate =
STenDim of int list * stimmediate list
| SExprDim of int * sexpr list



and sfunc_decl = {
   ftyp : t;
   sfname : string;
   sformals : (t * string) list;
   sbody : sstmt
}

and variable_decl = { (* Sast.Id wrapper*)
   vtyp  : t;
   sname : string;
   vexpr : sexpr;
}

and sstmt =
  SBlock of sstmt list
| Expr of sexpr
| If of sexpr * sstmt * sstmt
| For of sexpr * sexpr * sexpr * sstmt
| While of sexpr * sstmt
| Return of sexpr
| Callback of (Llvm.llbuilder -> Llvm.llbuilder)

(* program entry point *)
type sprogram = {
      functions : sfunc_decl list;
      globals : variable_decl list;
}

let get_line_sexpr = function
    Id(l, _, _) -> l
  | Binop(l, _, _, _, _) -> l
  | Unop(l, _, _, _) -> l
  | Assign(l, _, _, _) -> l
  | Call(l, _, _, _) -> l
  | Fliteral(l, _, _) -> l
  | Iliteral(l, _, _) -> l
  | Sliteral(l, _, _) -> l
  | TensorId(l, _, _, _) -> l
  | TensorAssign(l, _, _, _, _) -> l
  | Itensor(l, _, _) -> l
  | BindExpr(l, _, _) -> l
  | Noexpr(_) -> raise(Exceptions.NoExprHasNoLineNumberException)

let get_line_sbind = function
    InferredBind(l, _, _) -> l
```

```
  | TensorBind(l, _, _) -> l
  | TensorBindAndAssign(l, _, _, _) -> l

let rec string_of_sbind = function
    InferredBind(_, s, e) -> "let " ^ s ^ " = (" ^ string_of_sexpr e ^ ")"
  | TensorBind(_, s, q) -> "let " ^ s ^ "_{" ^ String.concat ", " (List.map string_of_sexpr q) ^
      "}"
  | TensorBindAndAssign(_, s, q, e) -> "let " ^ s ^ "_{" ^ String.concat ", " (List.map
      string_of_sexpr q) ^ "} = " ^
string_of_sexpr e

and string_of_stensor = function
   STenDim(dim, tl) -> "(" ^ String.concat " x "(List.map string_of_int dim) ^ ") [" ^
       String.concat ", " (List.map string_of_stensor tl) ^ "]"
   |SExprDim(i, el) -> "(" ^ string_of_int i ^ ") [" ^ String.concat ", " (List.map
       string_of_sexpr el) ^ "]"


and string_of_sexpr = function
    Id(_, t, s) -> (string_of_styp t) ^ " " ^ s
   |Noexpr(t) -> (string_of_styp t) ^ " Noexpr"
   |Binop(_, t, e1, op, e2) -> (string_of_styp t) ^ " (" ^ (string_of_sexpr e1) ^ ") " ^
       (Ast.string_of_op op) ^ " (" ^ (string_of_sexpr e2) ^ ")"
   |Unop(_, t, uop, e) -> (string_of_styp t) ^ " " ^ (Ast.string_of_uop uop) ^ " (" ^
       (string_of_sexpr e) ^ ")"
   |Assign(_, t, var, e) -> (string_of_styp t) ^ " " ^ var ^ " = (" ^ (string_of_sexpr e) ^ ")"
   |Call(_, t, fn, el) -> (string_of_styp t) ^ " " ^ (string_of_sfunc_call fn) ^ "(" ^
       String.concat "), (" (List.map string_of_sexpr el) ^ ")"
   |Fliteral(_, t, f) -> (string_of_styp t) ^ " " ^ string_of_float f
   |Iliteral(_, t, i) -> (string_of_styp t) ^ " " ^ string_of_int i
   |Sliteral(_, t, s) -> (string_of_styp t) ^ " " ^ s
   |TensorId(_, t, s, q) -> string_of_styp t ^ " " ^ s ^ "_{" ^ (String.concat ", " (List.map
       string_of_sexpr q)) ^ "}"
   |TensorAssign(_, t, s, sl, e) -> string_of_styp t ^ " " ^ s ^ "_{" ^ (String.concat ", "
       (List.map string_of_sexpr sl)) ^ ") =" ^ string_of_sexpr e
   |Itensor(_, t, i) -> string_of_styp t ^ " " ^ string_of_stensor i
   |BindExpr(_, t, b) -> (string_of_styp t) ^ " " ^ string_of_sbind b

and string_of_sfunc_call sfdecl =
   string_of_styp sfdecl.ftyp ^ " " ^
   sfdecl.sfname ^ "(" ^ String.concat ", " (List.map (fun n -> string_of_styp (fst n) ^ " " ^
       (snd n)) sfdecl.sformals) ^ "): " (* ): *)

and string_of_sfunc_decl sfdecl =
   string_of_styp sfdecl.ftyp ^ " " ^
   sfdecl.sfname ^ "(" ^ String.concat ", " (List.map (fun n -> string_of_styp (fst n) ^ " " ^
       (snd n)) sfdecl.sformals) ^
  ")\n" ^
  string_of_sstmt sfdecl.sbody ^
  "\n"

and string_of_sstmt = function
    SBlock(stmts) -> "{\n" ^ String.concat "" (List.map string_of_sstmt stmts) ^ "}\n"
  | Expr(e) -> "(" ^ string_of_sexpr e ^ ");\n"
  | Return(e) -> "return (" ^ string_of_sexpr e ^ ");\n"
```

```
134   | If(e, s, SBlock([])) -> "if (" ^ string_of_sexpr e ^ ")\n" ^ string_of_sstmt s
135   | If(e, s1, s2) -> "if (" ^ string_of_sexpr e ^ ")\n" ^ string_of_sstmt s1 ^ " else\n" ^
          string_of_sstmt s2
136   | For(e1, e2, e3, s) -> "for (" ^ string_of_sexpr e1 ^ " ; " ^ string_of_sexpr e2 ^ " ; " ^
          string_of_sexpr e3 ^ ") " ^ string_of_sstmt s
137   | While(e, s) -> "while (" ^ string_of_sexpr e ^ ") " ^ string_of_sstmt s
138   | Callback(_) -> ""
139
140
141 let string_of_sglobal global = string_of_styp global.vtyp ^ " " ^ global.sname ^ ";\n"
142
143 let string_of_program sast =
144   String.concat "" (List.map string_of_sglobal sast.globals) ^ "\n" ^
145   String.concat "\n" (List.map string_of_sfunc_decl sast.functions)
```

9.10 scanner.mll

```
{  open Parser
   open Lexing
   let filename = ref ""
   let lineno lb =
      lb.lex_curr_p.pos_lnum

  let unescape s =
    Scanf.unescaped s
 }

let alpha = ['a'-'z' 'A'-'Z']
let ascii = ([' '-'!' '#'-'[' ']'-'~'])
let digit = ['0'-'9']
let id = alpha (alpha | digit)*
(*Fix this junk for floating points.*)
let float = digit* ('.' digit*)? (['e' 'E'] ['-' '+']? digit+)?
let int = digit+
let whitespace = [' ' '\t' '\r']
let escape = '\\' ['\\' '\'' '"' 'n' 'r' 't']
let string = '"' ((escape | [^ '"' '\\'])* as str) '"'



rule token = parse
  [' ' '\t' '\r'] { token lexbuf } (* Whitespace *)
| '\n'     { Lexing.new_line lexbuf; token lexbuf } (* Newline *)
| "%{"     { comment lexbuf } (* Block comments *)
| "%"   { singlecomment lexbuf } (* Single-line comments *)
| ("%%" [^ '\n' ]*) as lxm { COMMENT(string_of_int lexbuf.lex_curr_p.pos_lnum, match String.length
    lxm with 0 | 1 | 2 -> "" | x -> String.sub lxm 2 (x-2)) }
| '('   { LPAREN }
| ')'   { RPAREN }
| '['   { LBRACKET }
| ']'      { RBRACKET }
| '{'   { LBRACE }
| '}'   { RBRACE }
| ';'   { SEMI }
```

```
| ','   { COMMA }
| '_'   { UNDER }
| '+'   { PLUS }
| '-'   { MINUS }
| '*'   { TIMES }
| '/'   { DIVIDE }
| '^'   { CONCAT }
| "=="    { EQ }
| '='   { ASSIGN }
| "~="    { NEQ }
| '<'   { LT }
| "<="    { LEQ }
| '>'   { GT }
| ">="    { GEQ }
| "&&"    { AND }
| "||"    { OR }
| '~'   { NOT }
| "else" { ELSE }
| "int"    { INT }
| "float"  { FLOAT }
| "void" { VOID }
| "for"    { FOR }
| "if"     { IF }
| "let"    { LET }
| "return" { RETURN }
| "string" { STRING }
| "while"  { WHILE }
| int as lxm { ILITERAL(string_of_int (lineno lexbuf), int_of_string lxm) }
| float as lxm { FLITERAL(string_of_int (lineno lexbuf), float_of_string lxm) }
| string {
   String.iter (fun n-> match n with '\n' -> Lexing.new_line lexbuf | _ -> ()) str;
   SLITERAL(string_of_int (lineno lexbuf), unescape str) }
| id as lxm { ID(string_of_int (lineno lexbuf), lxm) }
| eof { EOF }
| '"'      { raise (Exceptions.UnmatchedQuotationException(lineno lexbuf)) }
| _ as illegal { raise (Exceptions.IllegalCharacterException(lineno lexbuf, illegal)) }

and comment = parse
  "%}" { token lexbuf }
| '\n' { Lexing.new_line lexbuf; comment lexbuf } (* Increment line counter *)
| _ { comment lexbuf }

and singlecomment = parse
  ('\n') { Lexing.new_line lexbuf; token lexbuf } (* Single-line comment ends with a newline or a
      carriage return*)
| "%{" { comment lexbuf } (* If we see a multiline comment,
                          forget about looking for a newline *)
| _ { singlecomment lexbuf }
```

9.11 semant.ml

```
1 (*open Ast
2 open Sast*)
3
```

```
4
5  module StringMap = Map.Make(String)
6
7
8  (**** ENVIRONMENTS ****)
9  (* One environment is made for every scope. Environments link back to their parents so that
10    the chain of environments can be traversed to look for variables. Each environment
11    contains a scope-specific symbol table and the return type of the current function. *)
12 type env_func = {
13    func_type  : Sast.t;
14    func_args  : (string * Sast.t) list;
15 }
16
17 type env_var = {
18    var_type : Sast.t;
19    var_expr : Sast.sexpr;
20 }
21
22 type global_env = {
23    global_sym_table : env_var StringMap.t; (* for storing global variables *)
24    global_func_table : env_func StringMap.t (* stores functions: maps name -> type, list of
          function's formals types in order *)
25 }
26
27 type env = {
28    env_name : string;
29    env_sym_table  : env_var StringMap.t; (* map string -> Sast.t *)
30    env_parent  : env option; (* the global environment can have no parent *)
31   env_ret_typ : Sast.t;
32   func_block : bool
33 }
34
35 (**** HELPER FUNCTIONS ****)
36 (* Extract the type of a sexpr from the SAST *)
37 let extract_type sexpr_node = match sexpr_node with
38      Sast.Id(_, t, _)                  -> t
39    | Sast.Noexpr(t)                    -> t
40    | Sast.Binop(_, t, _, _, _)       -> t
41   | Sast.Unop(_, t, _, _)           -> t
42    | Sast.Assign(_, t, _, _)          -> t
43    | Sast.Call(_, t, _, _)             -> t
44   | Sast.Fliteral(_, t, _)         -> t
45    | Sast.Iliteral(_, t, _)           -> t
46    | Sast.Sliteral(_, t, _)           -> t
47    | Sast.TensorId(_, t, _, _)          -> t
48    | Sast.TensorAssign(_, t, _, _, _) -> t
49    | Sast.Itensor(_, t, _)             -> t
50    | Sast.BindExpr(_, t, _)           -> t
51
52 (* Convert an AST type into a SAST type *)
53 let convert_type = function
54      Ast.Void           -> Sast.Void
55    | Ast.Float           -> Sast.Float
56    | Ast.Int            -> Sast.Int
57    | Ast.String          -> Sast.String
58    | Ast.TensorType(q)  -> Sast.Tensor(List.length q)
```

```
59
60  (* Check that two given types are the same (or one is an int that can be promoted to float) *)
61  let check_matching line t1 t2 op = match t1, t2 with
62       x, y when x = y                    -> x
63     | Sast.Int, Sast.Float               -> Sast.Float
64     | Sast.Float, Sast.Int               -> Sast.Float
65     | Sast.Tensor(0), Sast.Tensor(0) | Sast.Tensor(0), Sast.Float | Sast.Float, Sast.Tensor(0) ->
         Sast.Float
66     | a, b                               -> raise (Exceptions.BinaryTypeMismatchException(line,
         (Sast.string_of_styp a), (Sast.string_of_styp b), op))
67
68  (*DEPRECATED Get the type of a given binary operator with the two given types *)
69  (*let get_btype t1 op t2 = match op with
70      Ast.Concat -> (match t1, t2 with
71          x, y when x != Sast.String || y != Sast.String ->
72             raise (Exceptions.BadConcatException(" TODO: linenum "))
73          |_                                -> Sast.String)
74     | Ast.Equal | Ast.Neq | Ast.Less | Ast.Leq | Ast.Greater | Ast.Geq | Ast.And | Ast.Or ->
         Sast.Int
75     |_ -> check_matching t1 t2 (Ast.string_of_op op)
76  *)
77
78  (* Attempt to find a given variable in the local symbol table(s), and then the global one *)
79  let rec try_find v local_env env =
80     try Some(StringMap.find v local_env.env_sym_table)
81         with Not_found -> (match local_env.env_parent with
82            None -> (try Some(StringMap.find v env.global_sym_table)
83               with Not_found -> None)
84      | Some x -> try_find v x env)
85
86
87
88  (**** CHECK EXPRESSIONS ****)
89  (* fold_left function to check the types of all expressions in one tensor dimension *)
90  let rec check_ten_expr_list a t = match a with
91     Sast.Tensor(r) -> if ((extract_type t) = a) then Sast.Tensor(r) else
         raise(Exceptions.InvalidTensorImmedException(Sast.get_line_sexpr t))
92     | Sast.Int | Sast.Float -> if ((extract_type t) == Sast.Float || (extract_type t) == Sast.Int)
         then Sast.Float else raise(Exceptions.InvalidTensorImmedException(Sast.get_line_sexpr t))
93     | _ -> raise(Exceptions.InvalidTensorImmedException(Sast.get_line_sexpr t))
94
95  (* extract the stimmediate from an expression we know is a tensor immediate *)
96  and convert_ten_to_stim t = match t with
97     Sast.Itensor(_,_, s) -> s
98     | x -> raise(Exceptions.InvalidTensorImmedException(Sast.get_line_sexpr x))
99
100 (* recursively check one tensor immediate *)
101 and check_ti line ti local_env global_env =
102    (* ti is a list of exprs. convert them to sexprs so we can decide if they are tensors or not *)
103    let tchecked = List.map (fun n -> check_expr n local_env global_env) ti in
104    (* check that everything in the same level of recursion is the same type. decide that this
         dimension is a dimension of floats or of tensors. *)
105    let tentype = List.fold_left check_ten_expr_list (extract_type (List.hd tchecked)) tchecked in
         match tentype with
106       (* if this dimension is made of tensors, extract the stimmediates *)
107       Sast.Tensor(_) -> let tstims = List.map convert_ten_to_stim tchecked in
```

```
108         (* take the first stimmediate and extract its list of dimensions *)
109         (let dim = extract_dim (List.hd tstims) in
110            (* make sure that all stimmediates in the list are of the same dimensions *)
111            List.iter (fun n -> let d = extract_dim n in
112            List.iter2 (fun a b -> if (a == b) then () else
                   (raise(Exceptions.InvalidTensorImmedException(line)))) d dim) tstims;
113            (* return (1) the new list of dimensions and (2) the list of tensorimmediates from this
                   level *)
114            Sast.STenDim((List.length tstims)::dim, tstims))
115       (* if this dimension is made of ints/floats, return a simple ExprDim *)
116       | Sast.Int | Sast.Float -> Sast.SExprDim(List.length tchecked, tchecked)
117       | _ -> raise(Exceptions.InvalidTensorImmedException(line))
118
119 and extract_dim t = match t with
120      Sast.SExprDim(i, _) -> [i]
121    | Sast.STenDim(il, _) -> il
122
123 and check_ten_expr expr local_env global_env =
124      let sexpr = check_expr expr local_env global_env in
125          (match extract_type sexpr with
126               Sast.Int | Sast.Float -> sexpr
127             | x -> raise( Exceptions.InvalidTensorElementException(Sast.get_line_sexpr sexpr,
                   Sast.string_of_styp x)))
128
129
130 (* Check one single index expression's type *)
131 and check_ind q local_env global_env =
132    let sq = check_ind_expr q local_env global_env in
133    match extract_type sq with
134         Sast.Void -> 1
135      | Sast.Int -> 0
136      | x -> raise(Exceptions.InvalidIndexException(Sast.get_line_sexpr sq, Sast.string_of_styp x))
137
138 (* Check the type of one index expression in a tensor *)
139 and check_ind_expr expr local_env global_env = match expr with
140   Ast.Id(line, vd) ->
141       (let vt = try_find vd local_env global_env in match vt with
142            None -> Sast.Id(line, Sast.Void, vd)
143          | Some x -> Sast.Id(line, x.var_type, vd))
144   | _ -> check_expr expr local_env global_env
145
146 (* Check a single expression. Check that types are proper, any identifiers can be found in the
147    symbol tables, and put any new declarations into them. Return a SAST.sexpr node. *)
148 and check_expr expr local_env global_env = match expr with
149     Ast.Id(line, vd) ->
150         (let vt = try_find vd local_env global_env in match vt with
151           None -> raise (Exceptions.NotDeclaredInScopeException(line, vd))
152         | Some x -> Sast.Id(line, x.var_type, vd))
153
154    |Ast.Noexpr -> Sast.Noexpr(Sast.Void)
155
156    |Ast.Binop(line, e1, op, e2) ->
157         let e1 = check_expr e1 local_env global_env
158         and e2 = check_expr e2 local_env global_env in
159        get_btype line e1 op e2
160
```

```
161     |Ast.Unop(line, op, e) ->
162         let e = check_expr e local_env global_env in
163     let t = extract_type e in
164     if (match op with
165            Ast.Neg -> t != Sast.Int && t != Sast.Float && t != Sast.Tensor(0)
166           |Ast.Not -> t != Sast.Int)
167         then raise (Exceptions.UnaryTypeMismatchException(line, (Sast.string_of_styp t),
                (Ast.string_of_uop op)))
168            else if t = Sast.Tensor(0) then Sast.Unop(line, Sast.Float, op, e)
169            else  Sast.Unop(line, t, op, e)
170
171     |Ast.Assign(line, s, e) ->
172          (let ts = try_find s local_env global_env in match ts with
173               None -> raise (Exceptions.NotDeclaredInScopeException(line, s))
174             | Some x -> let e = check_expr e local_env global_env in
175        Sast.Assign(line, check_matching line x.var_type (extract_type e) "assignment", s, e))
176
177     |Ast.Call(line, s, el) ->
178        let build_fname_from_call name elist local_env global_env = match name with
179           | _ -> name ^ "_" ^ (String.concat "_" (List.map (fun n -> Sast.string_of_styp(let t =
                extract_type(check_expr n local_env global_env) in (match t with Sast.Tensor(0) ->
                Sast.Float | x -> x))) elist))
180               in
181         let build_fnames name func = match name with
182            | _ -> List.map fst func.func_args
183            in
184         let fname = build_fname_from_call s el local_env global_env in
185         let func = (if ((String.length fname >= 11) && (String.compare (String.sub fname 0 4)
              "dim_") == 0 && (String.compare (String.sub fname ((String.length fname) - 11) 11)
              "-tensor_int") == 0) then StringMap.find "dim_0-tensor_int"
              global_env.global_func_table
186       else if ((String.length fname >= 7) && ((String.compare (String.sub fname 0 6) "print_") ==
            0) && ((String.compare (String.sub fname ((String.length fname) - 7) 7) "-tensor") ==
            0)) then StringMap.find "print_0-tensor" global_env.global_func_table
187       else (try StringMap.find fname global_env.global_func_table
188               with Not_found -> raise(Exceptions.NotDeclaredInScopeException(line, fname))))
189            in
190     let sformal_builder = List.map (fun n -> (match extract_type (check_expr n local_env
            global_env) with Sast.Tensor(0) -> Sast.Float | x -> x)) el in
191         let sformal_names = build_fnames s func in
192         Sast.Call(line, func.func_type, {Sast.ftyp = func.func_type; sfname = s; sformals
              =List.combine sformal_builder sformal_names; sbody = Sast.SBlock([])}, (List.map (fun
              n -> check_expr n local_env global_env) el))
193
194     |Ast.Fliteral(l,f) -> Sast.Fliteral(l, Sast.Float, f)
195
196     |Ast.Iliteral(l,i) -> Sast.Iliteral(l, Sast.Int, i)
197
198     |Ast.Sliteral(l,s) -> Sast.Sliteral(l, Sast.String, s)
199
200     |Ast.TensorId(line, s, q) ->
201         (let vt = try_find s local_env global_env in match vt with
202                  None -> raise (Exceptions.NotDeclaredInScopeException(line, s));
203               | Some x -> let rank = (List.fold_left (fun a n -> a + (check_ind n local_env
                    global_env)) 0 q) in
```

```
204       if (String.compare ((string_of_int (List.length q)) ^ "-tensor") (Sast.string_of_styp
              x.var_type)) != 0
205       then raise( Exceptions.TensorTypeException(line, s, Sast.string_of_styp x.var_type,
              string_of_int (List.length q)))
206       else Sast.TensorId(line, (if rank == 0 then Sast.Float else (Sast.Tensor(rank))), s, List.map
              (fun n -> check_ind_expr n local_env global_env) q))
207
208     |Ast.TensorAssign(line, s, sl, e) -> let e = check_expr e local_env global_env in
209         let id = check_expr (Ast.TensorId(line, s, sl)) local_env global_env in
210     let typ = (check_matching line (extract_type id) (extract_type e) "assignment") in
211          Sast.TensorAssign(line, typ, s, List.map (fun n -> check_ind_expr n local_env global_env)
                sl, e)
212
213     |Ast.Tensor(line, t) -> let stim = check_ti line t local_env global_env in
214                         let dim = extract_dim stim in
215                 Sast.Itensor(line, Sast.Tensor(List.length dim), stim)
216
217
218     |Ast.BindExpr(line, b) -> check_bind line b local_env global_env
219
220 and check_bind line bind local_env env = match bind with
221      Ast.InferredBind(l, s, e) -> check_new_id s l local_env;
222         let se = check_expr e local_env env in
223     let t1 = extract_type se in
224     let t = (match t1 with Sast.Tensor(0) -> Sast.Float
225              | x -> x) in
226             ignore(StringMap.add s {var_type=t; var_expr=se} local_env.env_sym_table);
227             Sast.BindExpr(line, t, Sast.InferredBind(line, s, se))
228     |Ast.TensorBind(line, s, q) -> let qsize = check_new_tensor s line q local_env in
229         ignore( List.map (fun n -> if (check_ind n local_env env) == 1 then
                raise(Exceptions.VoidTensorIndexException(line, Ast.string_of_expr n)) else ()) q );
230     if qsize == 0 then raise(Exceptions.EmptyTensorException(line))
231     else Sast.BindExpr(line, Sast.Tensor(qsize), Sast.TensorBind(line, s, List.map (fun n ->
            check_expr n local_env env) q))
232     |Ast.TensorBindAndAssign(line, s, q, e) -> let qsize = check_new_tensor s line q local_env in
233        let se = check_expr e local_env env in
234        ignore(check_matching line (extract_type se) (Sast.Tensor(qsize)) "assignment");
235        Sast.BindExpr(line, extract_type se, Sast.TensorBindAndAssign(line, s, List.map (fun n ->
               check_expr n local_env env) q, se))
236
237 and check_new_id id line local_env =
238     let x = StringMap.mem id local_env.env_sym_table in
239       match x with
240         true -> raise(Exceptions.DoubleDeclareException(line, id))
241       | false -> ()
242
243 and check_new_tensor s l q local_env =
244         check_new_id s l local_env;
245         let qsize = List.length q in
246     ignore(StringMap.add s {var_type=Sast.Tensor(qsize); var_expr=Sast.Noexpr(Sast.Void)}
            local_env.env_sym_table);
247     qsize
248
249
250 and extract_free_bound (f, b) q q2 =
251   if (extract_type q == Sast.Void) then (
```

```
252   if List.mem q q2 then (f, if not (List.mem q b) then List.append b [q] else b)
253     else (List.append f [q], b))
254   else (f, b)
255
256 and check_free_bound e1 e2 = (match e1 with
257   Sast.TensorId(_, _, _, q1) | Sast.TensorAssign(_, _, _, q1, _) ->
258     (match e2 with Sast.TensorId(_, _, _, q2) | Sast.TensorAssign(_, _, _, q2, _) ->
259       (let (free, bound) = List.fold_left (fun a n -> extract_free_bound a n q2) ([], []) q1 in
260       List.fold_left (fun a n -> extract_free_bound a n q1) (free, bound) q2)
261     | x -> raise (Exceptions.UnspecifiedTensorMultiplicationException(Sast.get_line_sexpr x)) )
262   | x -> raise (Exceptions.UnspecifiedTensorMultiplicationException(Sast.get_line_sexpr x)) )
263
264 (* Get the type of a given binary operator with the two given types *)
265 and get_btype line e1 op e2 =
266    let t1 = extract_type e1 in
267    let t2 = extract_type e2 in
268    match op with
269     Ast.Concat -> (match t1, t2 with
270         x, y when x != Sast.String || y != Sast.String ->
271           raise (Exceptions.BadConcatException(line))
272        |_                                -> Sast.Binop(line, Sast.String, e1, Ast.Concat, e2))
273    |Ast.Mult -> (match t1, t2 with
274    Sast.Tensor(_), Sast.Tensor(_) -> let (free, _) = check_free_bound e1 e2 in
275       Sast.Binop(line, Sast.Tensor(List.length free), e1, Ast.Mult, e2)
276       |Sast.Tensor(r), Sast.Float | Sast.Float, Sast.Tensor(r) -> Sast.Binop(line, Sast.Tensor(r)
              , e1, Ast.Mult, e2)
277       |Sast.Tensor(r), Sast.Int | Sast.Int, Sast.Tensor(r) -> Sast.Binop(line, Sast.Tensor(r) ,
              e1, Ast.Mult, e2)
278       |_, _ -> Sast.Binop(line, check_matching line t1 t2 (Ast.string_of_op op), e1, Ast.Mult, e2)
279 )
280    | Ast.And | Ast.Or | Ast.Less | Ast.Leq | Ast.Greater | Ast.Geq | Ast.Equal | Ast.Neq -> (match
          t1, t2 with
281         x, y when x == Sast.String || y == Sast.String ->
               raise(Exceptions.BinaryTypeMismatchException(line, (Sast.string_of_styp t1),
               (Sast.string_of_styp t2), (Ast.string_of_op op)))
282       | Sast.Tensor(_), _ -> raise(Exceptions.BinaryTypeMismatchException(line,
             (Sast.string_of_styp t1), (Sast.string_of_styp t2), (Ast.string_of_op op)))
283       | _, Sast.Tensor(_) -> raise(Exceptions.BinaryTypeMismatchException(line,
             (Sast.string_of_styp t1), (Sast.string_of_styp t2), (Ast.string_of_op op)))
284       | _, _ -> ignore(check_matching line t1 t2 (Ast.string_of_op op)); Sast.Binop(line,
             Sast.Int, e1, op, e2))
285    | x -> Sast.Binop(line, check_matching line t1 t2 (Ast.string_of_op op), e1, x, e2)
286
287 let extract_name_from_bind = function
288     Ast.InferredBind(_, s, _) -> s
289   | Ast.TensorBind(_, s, _) -> s
290   | Ast. TensorBindAndAssign(_, s, _, _) -> s
291
292 let extract_name_from_sbind = function
293     Sast.InferredBind(_, s, _) -> s
294   | Sast.TensorBind(_, s, _) -> s
295   | Sast.TensorBindAndAssign(_, s, _, _) -> s
296
297
298 (**** CHECK STATEMENT ****)
299 (* Check a single statement, Make a new environment if a new scope is started. Check all
```

```
300     expressions involved for correct types and semantic correctness. Return a SAST.sstmt
301     node. *)
302 let rec check_stmt local_env global_env stmt = match stmt with
303      Ast.Block(_,sl)          -> let new_env = if local_env.func_block == true then {local_env with
            func_block=false}
304                                          else {env_name="sblock"; env_sym_table=StringMap.empty;
                                                 env_parent=Some local_env;
                                                 env_ret_typ=local_env.env_ret_typ;
                                                 func_block=false}
305                                in check_sblock sl new_env global_env []
306    | Ast.Expr(_, e)        -> check_expr_stmt e local_env global_env
307    | Ast.If(_, e, s1, s2)    -> check_if e s1 s2 local_env global_env
308    | Ast.For(_, e1, e2, e3, s) -> check_for e1 e2 e3 s local_env global_env
309    | Ast.While(_, e, s)      -> check_while e s local_env global_env
310    | Ast.Return(_, e)         -> check_return e local_env global_env
311
312 (* check an expression statement *)
313 and check_expr_stmt e local_env global_env =
314    Sast.Expr(check_expr e local_env global_env)
315
316 and check_sexp_env e local_env global_env =
317   let checked_e = check_expr e local_env global_env in
318      match checked_e with
319         Sast.BindExpr(_, t, b) -> {local_env with env_sym_table=(StringMap.add
              (extract_name_from_sbind b) {var_type=t; var_expr=checked_e} local_env.env_sym_table)}
320       | _ -> local_env
321
322 and check_sblock sl local_env global_env outputlist = match sl with
323    [] -> if List.length outputlist > 0 && (String.compare local_env.env_name "function" == 0) then
          (let hd = List.hd outputlist in
324          (match hd with
325            Sast.Return(_) -> ()
326          | _ when local_env.env_ret_typ == Sast.Void-> ()
327          | _ -> raise(Exceptions.NoReturnException(Sast.string_of_styp local_env.env_ret_typ)));
328          Sast.SBlock(List.rev outputlist))
329          else Sast.SBlock(List.rev outputlist)
330   | Ast.Expr(_, e) :: tl -> (let new_env = check_sexp_env e local_env global_env in
331                           let sast_e = check_expr_stmt e local_env global_env in
332                             let updated_output_list = sast_e :: outputlist in
333                               check_sblock tl new_env global_env updated_output_list)
334   | Ast.Return(_, _) :: tl when tl != [] -> raise
         (Exceptions.StatementAfterReturnException(Ast.string_of_stmt (List.hd tl)))
335   | x :: tl -> let updated_output_list = check_stmt local_env global_env x :: outputlist in
336                check_sblock tl local_env global_env updated_output_list
337
338
339 (* check an if statement *)
340 and check_if e s1 s2 local_env global_env =
341    let exp = check_expr e local_env global_env in
342    let t = extract_type exp in
343    if t != Sast.Int then raise (Exceptions.ExpectedConditionException(Sast.get_line_sexpr exp,
          Ast.string_of_expr e));
344    let new_env_if = { env_name="if"; env_sym_table=StringMap.empty; env_parent=(Some local_env);
          env_ret_typ=local_env.env_ret_typ; func_block=false }
345    in let new_env_else = { new_env_if with env_name="else" } in
346    Sast.If(exp, check_stmt new_env_if global_env s1, check_stmt new_env_else global_env s2)
```

```
347
348 (* check a for statement *)
349 and check_for e1 e2 e3 s local_env global_env =
350     check_stmt local_env global_env (Ast.Block([], [Ast.Expr([],e1); Ast.While([],e2,
        Ast.Block([],[s; Ast.Expr([],e3)]))]))
351
352 (* check a while statement *)
353 and check_while e s local_env global_env =
354    let exp = check_expr e local_env global_env in
355    let t = extract_type exp in
356    if t != Sast.Int then raise (Exceptions.ExpectedConditionException(Sast.get_line_sexpr exp,
        Ast.string_of_expr e));
357    let new_env = { env_name="while"; env_sym_table=StringMap.empty; env_parent=(Some local_env);
        env_ret_typ=local_env.env_ret_typ; func_block=false } in
358    Sast.While(exp, check_stmt new_env global_env s)
359
360 (* check a return statement *)
361 and check_return e local_env global_env =
362     let exp = check_expr e local_env global_env in
363     if (String.compare (Sast.string_of_styp (extract_type exp)) (Sast.string_of_styp
         local_env.env_ret_typ)) != 0 then
364         raise (Exceptions.FuncReturnTypeException(Sast.get_line_sexpr exp, Sast.string_of_styp
             local_env.env_ret_typ, Sast.string_of_styp (extract_type exp)))
365     else Sast.Return(exp)
366
367
368 (**** GLOBALS ****)
369 (* check all global variables. put them into the global symbol table. *)
370 let check_globals globals global_env =
371     let fake_local_env = {env_name="foo"; env_sym_table=StringMap.empty; env_parent=None;
         env_ret_typ=Sast.Void; func_block=false } in
372     let check_gbind global_env bind = {global_env with global_sym_table = (
373        match bind with
374       Ast.InferredBind(line, s,e) ->
375          (let se = check_expr e fake_local_env global_env in
376           let t = extract_type se in
377                if StringMap.mem s global_env.global_sym_table
378                  then raise(Exceptions.DoubleDeclareException(line,s))
379                   else StringMap.add s {var_type=t;var_expr=se} global_env.global_sym_table)
380    | Ast.TensorBind(line, s, q) ->
381       ignore(check_new_id s line fake_local_env); (* check for redefinition *)
382       ignore(List.map (fun n -> if (check_ind n fake_local_env global_env) == 1 then
            raise(Exceptions.VoidTensorIndexException(line, Ast.string_of_expr n)) else ()) q);
383       let qsize = List.length q in
384       StringMap.add s {var_type=Sast.Tensor(qsize); var_expr=Sast.Noexpr(Sast.Void)}
            global_env.global_sym_table
385           |Ast.TensorBindAndAssign(line, s, q, e) ->
386       (ignore(check_new_id s line fake_local_env);
387        let se = check_expr e fake_local_env global_env in
388        let qsize = List.length q in
389        ignore(check_matching line (extract_type se) (Sast.Tensor(qsize)) "assignment");
390        StringMap.add s {var_type=Sast.Tensor(qsize); var_expr=se} global_env.global_sym_table)
391     )}
392     in List.fold_left check_gbind global_env globals
393
394
```

```
(**** FUNCTIONS ****)
(* Check one function. Make it its own local environment, check all its formals, check its body. *)
let check_function global_env dummy_env func =
   let local_env = {env_name="function"; env_sym_table=StringMap.empty; env_parent=dummy_env;
       env_ret_typ=(convert_type func.Ast.typ); func_block=true }
   in let add_formal local_env formal = { local_env with env_sym_table=
      if StringMap.mem (snd formal) local_env.env_sym_table
    then raise (Exceptions.DoubleDeclareException(" TODO: linenum", snd formal))
      else StringMap.add (snd formal) ({var_type= (convert_type (fst formal));
          var_expr=Sast.Noexpr(Sast.Void)}) local_env.env_sym_table }
   in let local_env_with_formals = List.fold_left add_formal local_env func.Ast.formals
   in let fsbody = check_stmt local_env_with_formals global_env func.Ast.body
   in let convert_formal formal = ((convert_type (fst formal)), (snd formal))
   in {Sast.ftyp=(convert_type func.Ast.typ); sfname=func.Ast.fname; sformals=List.map
       convert_formal func.Ast.formals; sbody=fsbody;}

(* Check all functions. For each function, add it to the global function table.
   Make sure some main() function is defined. *)
let rec add_to_globals global_env func = { global_env with global_func_table=(
  if StringMap.mem (build_fname_from_fdecl func.Ast.fname func.Ast.formals)
      global_env.global_func_table
       then raise (Exceptions.DoubleDeclareException(" TODO: linenum", func.Ast.fname))
  else StringMap.add (build_fname_from_fdecl func.Ast.fname func.Ast.formals)
      {func_type=convert_type func.Ast.typ; func_args=List.map (fun n -> ((snd n),(convert_type
      (fst n)))) func.Ast.formals } global_env.global_func_table )}
and build_fname_from_fdecl fn formals =
    fn ^ "_" ^ String.concat "_" (List.map (fun n-> Sast.string_of_styp (convert_type (fst n)))
        formals)

(* Check all functions, but also return the final global environment. *)
let check_functions functions dummy_env global_env =
   let new_global_env = List.fold_left add_to_globals global_env functions
   in let global_env_with_main = match StringMap.mem "main_" new_global_env.global_func_table with
      false -> raise (Exceptions.NoMainException)
         | true -> new_global_env
   in ((List.map (check_function global_env_with_main dummy_env) functions), global_env_with_main)


let built_in = [
(*file operations: fopen, fclose, fread, fwrite *)
              {Ast.comm=[]; Ast.typ=Ast.Int; Ast.fname="fopen"; Ast.formals=[(Ast.String, "f");
                  (Ast.String, "m")]; Ast.body=Ast.Block([],[])};
              {Ast.comm=[]; Ast.typ=Ast.Int; Ast.fname="fclose"; Ast.formals=[(Ast.Int, "f")];
                  Ast.body=Ast.Block([],[])};
              {Ast.comm=[]; Ast.typ=Ast.String; Ast.fname="fread"; Ast.formals=[(Ast.Int, "n");
                  (Ast.Int, "f")]; Ast.body=Ast.Block([],[])};
              {Ast.comm=[]; Ast.typ=Ast.Int; Ast.fname="fwrite"; Ast.formals=[(Ast.String, "s");
                  (Ast.Int, "f")]; Ast.body=Ast.Block([],[])};

(*print, toString, string length*)
              {Ast.comm=[]; Ast.typ=Ast.Int; Ast.fname="print"; Ast.formals=[(Ast.String, "s")];
                  Ast.body=Ast.Block([],[])};
              {Ast.comm=[]; Ast.typ=Ast.Int; Ast.fname="print"; Ast.formals=[(Ast.Int, "i")];
                  Ast.body=Ast.Block([],[])};
              {Ast.comm=[]; Ast.typ=Ast.Int; Ast.fname="print"; Ast.formals=[(Ast.Float, "f")];
                  Ast.body=Ast.Block([],[])};
```

```
437                 {Ast.comm=[]; Ast.typ=Ast.String; Ast.fname="toString"; Ast.formals=[(Ast.Int,
                        "i")]; Ast.body=Ast.Block([],[])};
438                 {Ast.comm=[]; Ast.typ=Ast.String; Ast.fname="toString"; Ast.formals=[(Ast.Float,
                        "f")]; Ast.body=Ast.Block([],[])};
439                 {Ast.comm=[]; Ast.typ=Ast.Int; Ast.fname="strlen"; Ast.formals=[(Ast.String, "s")];
                        Ast.body=Ast.Block([],[])};
440
441   (*tensor operations: dim, ... *)
442                   {Ast.comm=[]; Ast.typ=Ast.Int; Ast.fname="dim"; Ast.formals=[(Ast.TensorType([]),
                          "t"); (Ast.Int, "i")]; Ast.body=Ast.Block([], [])};
443                   {Ast.comm=[]; Ast.typ=Ast.Int; Ast.fname="print";
                          Ast.formals=[(Ast.TensorType([]), "t")]; Ast.body=Ast.Block([], [])};
444
445   (*casting*)
446                 {Ast.comm=[]; Ast.typ=Ast.Float; Ast.fname="itof"; Ast.formals=[(Ast.Int, "i")];
                        Ast.body=Ast.Block([],[])};
447                   {Ast.comm=[]; Ast.typ=Ast.Int; Ast.fname="ftoi"; Ast.formals=[(Ast.Float, "f")];
                          Ast.body=Ast.Block([], [])};
448                 {Ast.comm=[]; Ast.typ=Ast.String; Ast.fname="itos"; Ast.formals=[(Ast.Int, "i")];
                        Ast.body=Ast.Block([],[])};
449                   {Ast.comm=[]; Ast.typ=Ast.String; Ast.fname="ftos"; Ast.formals=[(Ast.Float,
                          "f")]; Ast.body=Ast.Block([], [])};
450                 {Ast.comm=[]; Ast.typ=Ast.Int; Ast.fname="stoi"; Ast.formals=[(Ast.String, "s")];
                        Ast.body=Ast.Block([],[])};
451                   {Ast.comm=[]; Ast.typ=Ast.Float; Ast.fname="stof"; Ast.formals=[(Ast.String,
                          "s")]; Ast.body=Ast.Block([], [])};
452
453   (*math*)      {Ast.comm=[]; Ast.typ=Ast.Float; Ast.fname="sqrt"; Ast.formals=[(Ast.Float, "f")];
       Ast.body=Ast.Block([], [])};
454
455                 ]
456
457   (* Add built-in functions to the global environment *)
458   let populate_library_functions global_env =
459       List.fold_left (fun g n -> add_to_globals g n) global_env built_in
460
461
462   (**** TYPIFY ****)
463   (* Take in a list of globals and a list of formals as defined in the AST. Semantically check all
464      of these, and if all goes well, return equivalent structures from the SAST. *)
465   let typify (globals, functions) =
466      (* use a dummy empty environment as the root to connect all other environments. the real global
467         vars/func decls will be stored in the global_env. *)
468      let env_root = { env_name = "root"; env_sym_table = StringMap.empty; env_parent = None;
           env_ret_typ=Sast.Void; func_block=false } in
469      let global_env = { global_sym_table=StringMap.empty; global_func_table=StringMap.empty } in
470      let global_env_with_globals = check_globals globals global_env in
471      let global_env_with_built_ins = populate_library_functions global_env_with_globals in
472      let (funcs, global_env_with_funcs) = check_functions functions (Some env_root)
           global_env_with_built_ins in
473      let convert_binding = function
474         | n, t -> { Sast.vtyp=t.var_type; vexpr=t.var_expr; sname=n }
475      in (global_env_with_funcs, {Sast.functions=funcs; Sast.globals=(List.map convert_binding
           (StringMap.bindings global_env_with_built_ins.global_sym_table)) })
476
477   let string_of_func func = (Sast.string_of_styp ((snd func).func_type)) ^ " " ^ (fst func) ^ " ("
```

```
478     ^ String.concat ", " (List.map (fun arg -> (Sast.string_of_styp (snd arg)) ^ " " ^ (fst arg))
          ((snd func).func_args)) ^ ");"
479
480 (* Pretty-print the program and also its global environment *)
481 let string_of_prog_and_env (global_env, sprogram) = Sast.string_of_program sprogram ^ "\n\n\nFINAL
      GLOBAL ENVIRONMENT:\n" ^
482    String.concat "" (List.map Sast.string_of_sglobal sprogram.Sast.globals) ^ "\n\n" ^
483    String.concat "\n" (List.map string_of_func (StringMap.bindings global_env.global_func_table))
```

9.12 stdlib.tens

```
1  float pow(float f, int i){
2       let result = f;
3
4       for(let j = 1; j<i; j = j+1){
5              result = result * f;
6       }
7
8       return result;
9  }
10
11 T_{a, b} transpose(T_{a, b} T) {
12      let da = dim(T, 0);
13      let db = dim(T, 1);
14
15      let newT_{da, db};
16      for(let i = 0; i<da; i = i+1){
17           for(let j = 0; j<db; j = j+1){
18      newT_{i, j}=T_{j,i};
19     }
20       }
21
22      return newT;
23 }
24
25
26 float min(T_{a} T) {
27     let da = dim(T, 0);
28     let min = T_{0};
29     for(let i = 1; i < da; i=i+1){
30       if(T_{i}<min) { min = T_{i}; }
31     }
32
33     return min;
34 }
35
36 int minInd(T_{a} T) {
37     let da = dim(T, 0);
38     let min = 0;
39     for(let i = 1; i < da; i=i+1){
40       if(T_{i}<T_{min}) { min = i; }
41     }
42
43     return min;
```

```
}

float max(T_{a} T) {
    let da = dim(T, 0);
    let max = T_{0};
    for(let i = 1; i < da; i=i+1){
      if(T_{i}>max) { max = T_{i}; }
    }

    return max;
}

int maxInd(T_{a} T) {
    let da = dim(T, 0);
    let max = 0;
    for(let i = 1; i < da; i=i+1){
      if(T_{i}>T_{max}) { max = i; }
    }

    return max;
}


float abs(float n){
      if(n<0){
         return -n;
      }
      return n;
}
```

9.13 testall.sh

```
#!/bin/sh

# Regression testing script for MicroC
# Step through a list of files
#  Compile, run, and check the output of each expected-to-work test
#  Compile and check the error of each expected-to-fail test

# Path to the LLVM interpreter
#LLI="lli"
LLI="/usr/local/opt/llvm/bin/lli"

# Path to the microc compiler. Usually "./microc.native"
# Try "_build/microc.native" if ocamlbuild was unable to create a symbolic link.
LATENS="./latens.native"
#LATENS="_build/latens.native"

# Set time limit for all operations
ulimit -t 30

globallog=testall.log
rm -f $globallog
error=0
```

```
globalerror=0

keep=0

Usage() {
    echo "Usage: testall.sh [options] [.tens files]"
    echo "-k   Keep intermediate files"
    echo "-h   Print this help"
    exit 1
}

SignalError() {
    if [ $error -eq 0 ] ; then
   echo "FAILED"
   error=1
    fi
    echo " $1"
}

# Compare <outfile> <reffile> <difffile>
# Compares the outfile with reffile. Differences, if any, written to difffile
Compare() {
    generatedfiles="$generatedfiles $3"
    echo diff -b $1 $2 ">" $3 1>&2
    diff -b "$1" "$2" > "$3" 2>&1 || {
   SignalError "$1 differs"
   echo "FAILED $1 differs from $2" 1>&2
    }
}

# Run <args>
# Report the command, run it, and report any errors
Run() {
    echo $* 1>&2
    eval $* || {
   SignalError "$1 failed on $*"
   return 1
    }
}

# RunFail <args>
# Report the command, run it, and expect an error
RunFail() {
    echo $* 1>&2
    eval $* && {
   SignalError "failed: $* did not report an error"
   return 1
    }
    return 0
}

Check() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
                          s/.tens//'`
    reffile=`echo $1 | sed 's/.tens$//'`
```

```
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

    echo 1>&2
    echo "###### Testing $basename" 1>&2

    generatedfiles=""

    generatedfiles="$generatedfiles ${basename}.ll ${basename}.out" &&
    Run "$LATENS" "<" $1 ">" "${basename}.ll" &&
    Run "$LLI" "${basename}.ll" ">" "${basename}.out" &&
    Compare ${basename}.out ${reffile}.out ${basename}.diff

    # Report the status and clean up the generated files

    if [ $error -eq 0 ] ; then
   if [ $keep -eq 0 ] ; then
      rm -f $generatedfiles
   fi
   echo "OK"
   echo "###### SUCCESS" 1>&2
    else
   echo "###### FAILED" 1>&2
   globalerror=$error
    fi
}

CheckFail() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
                          s/.tens//'`
    reffile=`echo $1 | sed 's/.tens$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

    echo 1>&2
    echo "###### Testing $basename" 1>&2

    generatedfiles=""

    generatedfiles="$generatedfiles ${basename}.err ${basename}.diff" &&
    RunFail "$LATENS" "<" $1 "2>" "${basename}.err" ">>" $globallog &&
    Compare ${basename}.err ${reffile}.err ${basename}.diff

    # Report the status and clean up the generated files

    if [ $error -eq 0 ] ; then
   if [ $keep -eq 0 ] ; then
      rm -f $generatedfiles
   fi
   echo "OK"
   echo "###### SUCCESS" 1>&2
    else
   echo "###### FAILED" 1>&2
```

```
   globalerror=$error
    fi
}

while getopts kdpsh c; do
    case $c in
   k) # Keep intermediate files
       keep=1
       ;;
   h) # Help
       Usage
       ;;
    esac
done

shift `expr $OPTIND - 1`

LLIFail() {
  echo "Could not find the LLVM interpreter \"$LLI\"."
  echo "Check your LLVM installation and/or modify the LLI variable in testall.sh"
  exit 1
}

which "$LLI" >> $globallog || LLIFail


if [ $# -ge 1 ]
then
    files=$@
else
    files="tests/test-*.tens tests/fail-*.tens"
fi

for file in $files
do
    case $file in
   *test-*)
       Check $file 2>> $globallog
       ;;
   *fail-*)
       CheckFail $file 2>> $globallog
       ;;
   *)
       echo "unknown file type $file"
       globalerror=1
       ;;
    esac
done

exit $globalerror
```

9.14 script_num_pass.sh

```
#!/bin/bash
```

```
cat testall.log |tr " " "\n"|sort|uniq -c |sort -n -r -k 1 |grep -w -f list.txt
```

9.15 list.txt

```
FAILED
SUCCESS
```

9.16 Demos

9.16.1 demo0.tens

```
    int fib(int n) {
   if (n < 2) {
      return 1;
   }

   return fib(n - 1) + fib(n - 2);
}

int main () {
   %% Declare a 1-dimensional tensor.
   let T = [2, 4, 8, 16, 32, 64];

   print("Hello " ^ "world!\n");
   print("\n");


   %% For loop. Print all elements of a tensor.
   for (let i = 0; i < dim(T, 0); i = i + 1) {
      print("Element " ^ itos(i) ^ " of tensor T is " ^ ftos(T_{i}) ^ ".\n");
   }
   print(ftos(max(T)) ^ " is the maximum, " ^ ftos(min(T)) ^ " is the minimum.");
   print("\n");


   let n = 6;
   let prod = 1;

   %% While loop. Iterative factorial.
   while (n > 0) {
      prod = prod * n;
      n = n - 1;
   }

   print("6! is " ^ itos(prod) ^ ".\n");
   print("\n");


   let k = 5;
   let f = fib(k);

```

```
    print("The " ^ itos(k + 1) ^ "th fibonacci number is " ^ itos(f) ^ ".\n");
    print("\n");


    let a = -4;
    let b = 13.6;
    let c = a + b;

    print("Variable a is an int: " ^ itos(a) ^ ", and variable b is a float: " ^ ftos(b) ^ ", but a
        + b is a float: " ^ ftos(c));
    print("\n");


    let W = [ 03, 2, 54/7+3, n, -44, 42];
    print("W is ");
    print(W);
    print("\nW+T is ");
    print(W+T);

    % tensors: tensor mult and tensor add
    let y = W_{z}*T_{z};
    print("T dot W is: " ^ ftos(y) ^ "\n");

    % file i/o

    let file = fopen("test","w");
    let message = "Hello, World!\nCruel Cruel World!";
    fwrite(message, file);
    fclose(file);

    file = fopen("test","r");
    let len = strlen(message) - 1.;
    message = fread(ftoi(len), file);
    message = message ^ " :)";
    print(message ^ "\n");
    fclose(file);


    return 0;
}
```

9.16.2 demo1.tens

```
    %% A pithy code sample that shows us
%% how much cattle we would have if we
%% bought 40 cattle some number of years ago.
%% With the help of our friends: https://www.wku.edu/mathmatters/documents/mathmattersep13.pdf
%% (ie we implemented their stuff in LaTenS)
int main () {
    print("Suppose you purchase 40 head of 2 year old cattle\n");

    %% Our Leslie Matrix, for heads of cattle
    let L = [[ 0, .5, 1.1, 1.3, 0.8, .4],
             [.7, 0, 0, 0, 0, 0],
```

```
                [0, .9, 0, 0, 0, 0],
                [0, 0, .8, 0, 0, 0],
                [0, 0, 0, .7, 0, 0],
                [0, 0, 0, 0, .3, 0]];

    %% We've got 40 cattle at t=0; where t is measured in years
    let D0 = [0, 40, 0, 0, 0, 0];


    print("given survival rates and birth rates: \n");
    print(L);


    print("you want to know what your age distribution is\n");
    print("after 2 years: ");

    %% Each multiplication moves t forward 2 years.
    %% So now t=2
    let D = L_{a, b}*D0_{b};
    print(D);


    %% Let's jump t forward 28 more years
    %% So t = 2 + 28 = 30
    for(let i = 0; i < 14; i = i+1){
      D = L_{a, b}*D_{b};
    }
    print("after 30 years: ");
    print(D);


    %% And of course, no demo would be complete
    %% without showing t=42
    for(let i = 0; i < 26; i = i+1){
      D = L_{a, b}*D_{b};
    }
    print("after 42 years: ");
    print(D);

    return 0;
}
```

9.16.3 demo2.tens

```

       let M = 5.972e24;
    let G = 6.67408e-11;
    let c = 2.99e8;
    let v = 3874.;
    let re = 6.357e6;
    let o = 2.6541e7;
    let sperday = 86400.;


```

```
int main () {
   let r = re+o;

   let S_{4, 4};
   for(let i = 0; i<4; i = i+1){
      for(let j = 0; j<4; j=j+1){
      S_{i, j} = 0;
   }
    }

    let x=2*G*M/(r*pow(c, 2));

    S_{1, 1} = 1/sqrt(1-x); %1+x/2+3/8*pow(x, 2)+5/16*pow(x, 3);
    S_{0, 0} = -1/S_{1,1};
    S_{2, 2} = S_{3, 3} = -r;

   print("\nOver the course of one day, ");


   getDilation(sperday, "day");
   print("s/day\n");


   print("over the course of one week, ");

   getDilation(sperday*7, "week");


   print("over the course of 42 days, ");

   getDilation(sperday*42, "42 days");


    return 0;
}

void getDilation(float t, string name){
    let g = v*v/(c*c);
    let gg = G*M/(re*pow(c, 2))-G*M/(o*pow(c, 2));

    let ts = t*(1.+g/2.+3./8.*pow(g, 2)+5./16.*pow(g, 3)) -t;
    let tg = t*gg;



   print("you lose " ^ ftos(ts) ^ " s from special relativity and gain " ^ ftos(tg) ^ " from
       general relativity\n");

   let err = -ts+tg;
   print(err);
   print("so a gps clock would find " ^ ftos(t+err) ^ "s/" ^ name);
   print(" which results in an error of " ^ ftos(err*c) ^ "m/" ^ name ^ "\n");


}
```

9.16.4 demo3.tens

```
    %% Perceptron
int main() {
   let trainingData = [[0, 0, 0, 0, 1],
            [0, 0, 0, 1, 1],
            [0, 0, 1, 0, 1],
            [0, 0, 1, 1, 1],
            [0, 1, 0, 0, 1],
            [0, 1, 0, 1, 1],
            [0, 1, 1, 0, 1],
            [0, 1, 1, 1, 1],
            [1, 0, 0, 0, 1],
            [1, 0, 0, 1, 1],
            [1, 0, 1, 0, 1],
            [1, 0, 1, 1, 1],
            [1, 1, 0, 0, 1],
            [1, 1, 0, 1, 1],
            [1, 1, 1, 0, 1],
            [1, 1, 1, 1, 1]];
   let trainingLabels = [-1, -1, -1, 1, -1, -1, -1, 1, -1, -1, -1, 1, 1, 1, 1, 1];


   let theta_{dim(trainingData, 1)};

   let check = 1; % check if the classifier makes any mistake
   let count = 0; % returns the number of iterations needed to perfectly classify linearly
       separable data (or how many iterations we tried)
   let mistakes = 0;

   %% Halt after 500 iterations over the dataset
   while (check == 1 && count < 500) {
      check = 0;
      count = count + 1;

      %% Loop through the entire dataset
      for (let i = 0; i < dim(trainingLabels, 0); i = i + 1) {

         let y = trainingLabels_{i};
         let x = trainingData_{i,a};
         % the first row of the matrix dotted with theta
         let guess = x_{a} * theta_{a};

         %% If the classifier has made a mistake, update it
         if (guess * y <= 0)
         %% Update the classifier
         {
            check = 1;
            mistakes = mistakes + 1;
            %% For each element of theta, add y*x_i
            for (let j = 0; j < dim(theta, 0); j = j + 1) {
               theta_{j} = theta_{j} + (y * x_{j});
            }
         }
      }
   }
```

```
54
55    print("FINAL CLASSIFIER:\n");
56    print(theta);
57    print(itos(mistakes) ^ " mistakes made over " ^ itos(count) ^ " iterations.\n");
58
59    return 0;
60 }
```

9.17 Tests

9.17.1 fail-add.err

```
1 Line: 5: Operator + does not support types int and string.
```

9.17.2 fail-add.tens

```
1 int main () {
2    let a = 3;
3    let b = "hello";
4
5    let c = a + b; %{ error: cannot add int and string %}
6
7    return 0;
8 }
```

9.17.3 fail-and.err

```
Line: 5: Operator && does not support types string and string.
```

9.17.4 fail-and.tens

```
1 int main () {
2     let a = "LaTenS is";
3     let b = "the very bestest!";
4
5     print(a && b); %{ error: can't and tensors %}
6
7     return 0;
8 }
```

9.17.5 fail-assign1.err

```
Line: 9: Operator assignment does not support types string and int.
```

9.17.6 fail-assign1.tens

```
int main () {
   let a = 0;
   let b = 3.6;

   b = a; %{ this is fine; a will get promoted to float %}

   let c = "hello world";

   c = a; %{ not okay: cannot assign int to string %}

   return 0;
}
```

9.17.7 fail-assign2.err

```
    Line: 1: Char 13 - Line: 1 Char 13: syntax error
```

9.17.8 fail-assign2.tens

```
int main () {
   let a = if (0) 2; else 100; %{ statements cannot be used on the right hand side of an
       assignment %}

   print(a);

   return 0;
}
```

9.17.9 fail-codeoutsidefunc.err

```
Line: 7: Char 1 - Line: 7 Char 1: syntax error
```

9.17.10 fail-codeoutsidefunc.tens

```
int add(int a, int b) {
   return a + b;
}

int main () {
   return 0;
}

%{ we do not allow any code outside a function %}
add(1, 2);
```

9.17.11 fail-comments.err

```
Line: 1: Char 12 - Line: 1 Char 12: syntax error
```

9.17.12 fail-comments.tens

```
int main() {
   let a;
   let b;

   %{ this is a comment missing an end brace. oops. %

   a = 1.3;
   let c = a * a;

   return 0;
}
```

9.17.13 fail-declare-tensor1.err

```
Line: 2: Invalid tensor immediate.
```

9.17.14 fail-declare-tensor1.tens

```
int main () {
   let T_{a} = [1, 2.0, "hello"]; %{ ints and floats are okay inside a tensor (all promoted to
       floats), but srings are not %}

   return 0;
}
```

9.17.15 fail-declare-tensor2.err

```
Line: 2: Invalid tensor immediate.
```

9.17.16 fail-declare-tensor2.tens

```
int main () {
   let T_{a, b} = [[1, 2], [1, 2, 3]]; % the subarrays inside the tensor are of different sizes

   return 0;
}
```

9.17.17 fail-declare1.err

```
    Line: 1: Char 13 - Line: 1 Char 13: syntax error
```

9.17.18 fail-declare1.tens

```
    int main () {
   let my_var = 3.0; %{ underscore is reserved; should error %}

   return 0;
}
```

9.17.19 fail-declare2.err

```
Line: 1: Char 13 - Line: 1 Char 13: syntax error
```

9.17.20 fail-declare2.tens

```
int main () {
   let while = "hello"; %{ while is a reserved word %}

   return 0;
}
```

9.17.21 fail-declare3.err

```
    Line: 2: Illegal character: \'
```

9.17.22 fail-declare3.tens

```
    int main () {
   let a = 'This is an invalid string!'; %{ String cannot be declared with single quotes %}

   return 0;
}
```

9.17.23 fail-declare4.err

```
Line: 4: T already declared in current scope.
```

9.17.24 fail-declare4.tens

```
int main () {
   let T = [1, 2, 3];

   let T = 4; %{ this is redeclaring a variable since they have the same name %}

   return 0;
}
```

9.17.25 fail-expr.err

```
Line: 5: Char 7 - Line: 5 Char 8: syntax error
```

9.17.26 fail-expr.tens

```
int main () {
   let a = 3;
   let b = 4;

   print(a == ); %{ we should be able to catch a badly-formed expression %}

   return 0;
}
```

9.17.27 fail-func1.err

```
Line: 1: Char 16 - Line: 3 Char 10: syntax error
```

9.17.28 fail-func1.tens

```
int fib(int n) {
   fib0 = 1;
   fib1 = 1;

   %{ error: can't define a function inside another function %}
   int fibloop(int fib1, int fib2) {
      return fib1 + fib2;
   }

   for (i = 2; i <= 2; i = i + 1) {
      int tmp = fib0;
      fib0 = fib1;
      fib1 = fibloop(tmp, fib1);
   }

   return 0;
}
```

```

int main () {
   print(fib(5));
}
```

9.17.29 fail-func2.err

```
Expected return statement of type: string but no return statement found.
```

9.17.30 fail-func2.tens

```
string println(int i) {
   print(i);
}

%{ can't declare two fnctions with the same name even if they have different return types or args
    %}
string println(float f) {
   print(f);
}

int main () {
   println(4);
   println(-1.6e1);
}
```

9.17.31 fail-func3.err

```
Expected return statement of type: int but no return statement found.
```

9.17.32 fail-func3.tens

```
%{ this function should have a return statement %}
int add(int a, int b) {
   let c = a + b;
}

int main () {
   print(add(1, 2));

   return 0;
}
```

9.17.33 fail-func4.err

```
Line: 3: Expected return type of string but found float instead.
```

9.17.34 fail-func4.tens

```
%{ the function is declared as a string but returns a float %}
string helloworld() {
   return 1.0;
}

int main () {
   print(helloworld());

   return 0;
}
```

9.17.35 fail-func5.err

```
Line: 5: Expected return type of void but found int instead.
```

9.17.36 fail-func5.tens

```
%{ void function shouldn't return anything %}
void sayhello() {
   print("hello world!");

   return 1;
}

int main () {
   sayhello();

   return 0;
}
```

9.17.37 fail-geq.err

```
Line: 5: Operator >= does not support types 1-tensor and 1-tensor.
```

9.17.38 fail-geq.tens

```
int main () {
   let T1 = [9, 9, 9];
   let T2 = [3, 3, 3];

   print(T1 >= T2); %{ can't compare tensors with >= %}

   return 0;
}
```

9.17.39 fail-gt.err

```
Line: 5: Operator > does not support types string and string.
```

9.17.40 fail-gt.tens

```
int main () {
   let str1 = "aaa";
   let str2 = "jjj";

   print(str1 > str2); %{ error: can't compare strings with greater than %}

   return 0;
}
```

9.17.41 fail-leq.err

```
Line: 5: Operator <= does not support types string and string.
```

9.17.42 fail-leq.tens

```
int main () {
   let s1 = "test";
   let s2 = "Test";

   print(s1 <= s2); %{ error: can't compare strings with <= %}

   return 0;
}
```

9.17.43 fail-lt.err

```
Line: 5: Operator < does not support types 1-tensor and 1-tensor.
```

9.17.44 fail-lt.tens

```
int main () {
   let T1 = [1, 1, 1];
   let T2 = [2, 2, 2];

   let c = T1 < T2; %{ error: can't compare tensors with < %}

   return 0;
}
```

9.17.45 fail-nobraces.err

```
Line: 1: Char 13 - Line: 1 Char 13: syntax error
```

9.17.46 fail-nobraces.tens

```
int main () {
   let i;
   let a = 0;

   for (i = 0; i < 10; i = i + 1) {
      a = a + 10;
   %{ error: should understand braces are mismatched %}

   return 0;
}
```

9.17.47 fail-nomain.err

```
LaTenS files need a main method.
```

9.17.48 fail-nomain.tens

```
int foo (int a) {
   return a + 1;
}

float bar (float b) {
   return b * 1.5;
}

%{ error: no main function %}
```

9.17.49 fail-noparens.err

```
Line: 8: Char 13 - Line: 8 Char 62: syntax error
```

9.17.50 fail-noparens.tens

```
int foo (int b) {
   return b * 10;
}

int main () {
   let T1_{a,b} = [[1, 0], [0, 1]];

```

```
8     let a = foo(((T1_{0,0} * (3 + 2)) + (T1_{1,(0 - 0)} + 4) / 2); %{ there is one missing paren
          after the 4 %}
9
10    return 0;
11 }
```

9.17.51 fail-nosemi.err

```
Line: 6: Char 1 - Line: 6 Char 25: syntax error
```

9.17.52 fail-nosemi.tens

```
1  int main () {
2     let a = 4;
3     let b = 2.6;
4     let c_{a} = [1, 2, 3, 4];
5
6     print(a + b * a - c_{2}) %{ missing semicolon at the end of a statement %}
7
8     return 0;
9  }
```

9.17.53 fail-or.err

```
Line: 5: Operator || does not support types string and string.
```

9.17.54 fail-or.tens

```
1  int main () {
2     let s1 = "string";
3     let s2 = "string 2";
4
5     print(s2 || s2); %{ error: can't or strings %}
6  }
```

9.17.55 fail-strcat.err

```
Line: 5: Concatenation can only be used with strings.
```

9.17.56 fail-strcat.tens

```
1  int main () {
2     let s = "hello ";
3     let f = -1.3e4;
4
```

```
    let cat = s ^ f; %{ error: can't concat strings and floats...unless we want to cast to string
        automatically? we have the str() function though) %}

    return 0;
}
```

9.17.57 fail-sub.err

```
Line: 5: Operator - does not support types float and string.
```

9.17.58 fail-sub.tens

```
int main () {
    let f = +1.26;
    let s = "testing";

    let test = f - s; %{ error: can't subtract strings %}

    return 0;
}
```

9.17.59 fail-tensor-add.err

```
Line: 2: Operator + does not support types 2-tensor and 3-tensor.
```

9.17.60 fail-tensor-add.tens

```
int main () {
    print([[1.0, 2], [1.2, 2.0]] + [[[1.1, 2], [3.23, 4]], [[1.0, 2], [3, 4.42]], [[1.63, 2.8], [3,
        4]]]);
    % error: shouldn't be able to add tensors of different rank

    return 0;
```

9.17.61 fail-tensor-negate.err

```
Line: 3: Type 1-tensor not supported by op: -.
```

9.17.62 fail-tensor-negate.tens

```
int main() {
    let A = [3,4];
    print(-A);

    return 0;
```

```
}
```

9.17.63 fail-sub2.err

```
Line: 2: Operator - does not support types 1-tensor and 2-tensor.
```

9.17.64 fail-sub2.tens

```
int main () {
   print([1, -2.0, 3.0] - [[1.0, 2, 3.21], [1.2, 2.22, 3]]);
   % can't subtract tensors of different rank

   return 0;
}
```

9.17.65 fail-uminus.err

```
Line: 4: Type string not supported by op: -.
```

9.17.66 fail-uminus.tens

```
int main () {
   let str = "test string";

   print(-str); %[ error: can't do unary minus on string %}

   return 0;
}
```

9.17.67 test-assign.out

```
15.000000
42
```

9.17.68 test-assign.tens

```
int main () {
   let a = -0.1e-5;
   let b = 43;
   a = 8.4 + 2.2;
   print(a + 4.4);
   print(b - 1);

   return 0;
}
```

9.17.69 test-comments1.out

```
test
```

9.17.70 test-comments1.tens

```
% testing single-line comments

% single-line comments all over the file should be ignored
% by the scanner
% and everything else should compile properly
int main () {
   % print("hello!");
   print("test");
   % print ("hello world!");

   return 0;
}
```

9.17.71 test-comments2.out

```
test
test
```

9.17.72 test-comments2.tens

```
%{ testing multi-line comments %}

%{ testing longer multiline comments
   that use multiple lines %}
int main () {
   %{ test
   test
   test
   print("hello!\n");
   test
   %}

   print("test\n");

   print("test" %{ + "ing multiline comments" %} );

   return 0;
}
```

9.17.73 test-comments3.out

```
test
```

9.17.74 test-comments3.tens

```
%{ % test combinung the types of comments %}

int main () {
   %{ %testing that the % sign inside a multiline
      %comment doesn't do anything
      strange to the comment rule
      print("hello");
   %}

   % and vice-versa checking that we can %{
     identify a multi-line comment starting
     inside a single-line comment
     print("hello"); %}

   print("test");

   return 0;
}
```

9.17.75 test-declare.out

```
   7
-2
10.500000
16.715400
-12.761000
31000.000000
-0.000743
hello
"hello,
 world!"
1.000000
1.000000
-1.000000
```

9.17.76 test-declare.tens

```
    int main () {
        let c = 7; % ints
        let d = -2;
        let e = 10.5; % floats
        let f = 16.71540;
        let g = -12.761;

        let h = 31e3;
        let i = -743.265e-6;
        let j = "hello\n"; % strings
        let k = "\"hello, \n world!\"\n";
        let l = [1, 2, 3, 4]; % tensors
        let m = [[1, 0, 0], [0, 1, 0], [0, 0, 1]];
```

```
        let n = [-1, 3.2, 0.001, 7.8, 4];

        print(c);
        print(d);
        print(e);
        print(f);
        print(g);
        print(h);
        print(i);
        print(j);
        print(k);
        print(l_{0});
        print(m_{0,0});
        print(n_{0});

        return 0;
}
```

9.17.77 test-div.out

```
   1.000000
4.500000
```

9.17.78 test-div.tens

```
    int main () {
        let T = [[1, 2], [3.2, 4.5]] / [[1.0, 1], [1, 1.0]]; %{ we do not allow tensor division %}

        print(T_{0, 0});
        print(T_{1, 1});

        return 0;
}
```

9.17.79 test-float-ops1.out

```
   42.000000
4200.000000
0.420000
-4200.000000
-0.042000
```

9.17.80 test-float-ops1.tens

```
    int main(){
    print(42.);
    print(42e2);
    print(42e-2);
    print(-42e2);
```

```
    print(-42e-3);

    return 0;
}
```

9.17.81 test-float-ops2.out

```
   49.180000
33.580000
43.594200
4.600000
4.000000
-25.150000
25.150000
19.310000
0
1
0
0
1
1
1
0
0
1
0
1
```

9.17.82 test-float-ops2.tens

```
int main () {
      print(12.27 + 36.91);
      print(43.15 - 9.57);
      print(3.51 * 12.42);
      print(23.0 / 5.0);
      print(20.0 / 5.0);
      print(-25.15);
      print(--25.15);
      print(+19.31);
      print(54.0 > 54.01);
      print(21.0 > 20.99);
      print(53.0 > 53.0);
      print(14.0 >= 14.01);
      print(85.0 >= 84.99);
      print(14.0 >= 14.0);
      print(76.0 < 76.01);
      print(12.0 < 11.99);
      print(84.0 < 84.0);
      print(17.0 <= 17.01);
      print(19.0 <= 18.99);
      print(28.0 <= 28.0);

```

```
        return 0;
}
```

9.17.83 test-for1.out

```
 0
1
2
3
4
done
```

9.17.84 test-for1.tens

```
    int main () {
        for(let i = 0; i < 5; i = i + 1) {
               print(i);
        }

        print("done");

        return 0;
}
```

9.17.85 test-for2.out

```
 0
1
2
3
4
done
```

9.17.86 test-for2.tens

```
int main () {
   let i = 0;

   for ( ; i < 5 ; ) {
      print(i);
      i = i + 1;
   }

   print("done");
   return 0;
}
```

9.17.87 test-for3.out

```
0
0
1
2
3
4
1
0
1
2
3
4
2
0
1
2
3
4
3
0
1
2
3
4
4
0
1
2
3
4
done
```

9.17.88 test-for3.tens

```
int main () {

   for (let i = 0; i < 5; i = i + 1) {
      print(i);

      for (let j = 0; j < 5; j = j + 1) {
         print(j);
      }
   }

   print("done");
   return 0;
}
```

9.17.89 test-for4.out

```
done
```

9.17.90 test-for4.tens

```
int main () {

    for(;0;){
   print("this shouldn't print");
    }

    print("done");
    return 0;
}
```

9.17.91 test-func1.out

```
15
```

9.17.92 test-func1.tens

```
int foo (int bar) {
   return bar + 10;
}

int main () {
   print(foo(5));

   return 0;
}
```

9.17.93 test-func2.out

```
1
2.000000, 2.000000,
1.000000, 1.500000,
```

9.17.94 test-func2.tens

```
int foo () {
   return 1;
}

T_{a,b} bar () {
   return [[2, 2], [1, 1.5]];
}

int main () {
```

```
   print(foo());
   print(bar());


   return 0;
}
```

9.17.95 test-func3.out

```
4
1
hello
```

9.17.96 test-func3.tens

```
int foo(int a) {
   print(a);
   return a;
}

int bar(int b) {
   print(b + 1);
   return b + 1;
}

void baz(int a, int b) {
   print("hello\n");
}

int main() {
   baz(foo(1), bar(3));
   return 0;
}
```

9.17.97 test-func4.out

```
hello world!
```

9.17.98 test-func4.tens

```
void printSpecial(string s) {
   print(s);
}

int main () {
   let str = "hello world!";
   printSpecial(str);

   return 0;
```

```
}
```

9.17.99 test-func5.out

```
5
4
3
2
1
120
```

9.17.100 test-func5.tens

```
int factorial (int n) {
   print(n);
   if((n == 1) || (n == 0)) { return 1; }

   return n * factorial(n - 1);
}

int main () {
   print(factorial(5));

   return 0;
}
```

9.17.101 test-if1.out

```
true
true
done
```

9.17.102 test-if1.tens

```
int main () {
   if (1) print("true\n");
   if (0) print("false\n");

   if (1 || 0) {
      print("true\n");
   }

   print("done");

   return 0;
}
```

9.17.103 test-if2.out

```
true
true
```

9.17.104 test-if2.tens

```
int main () {
   let t = 1;
   let f = 0;

   if (t) print("true\n");
   else print("false\n");

   if (f) {
      print("false\n"); }
   else {
      print("true\n");
   }

   return 0;
}
```

9.17.105 test-if3.out

```
true
```

9.17.106 test-if3.tens

```
    int main () {
        if (0) print("false");
        else if (1) print("true");

        return 0;
    }
```

9.17.107 test-if4.out

```
    false
```

9.17.108 test-if4.tens

```
    int main() {
        if (0) {
                print("true");
        } else if (0 || 0) {
```

```
5               print("true");
6       } else {
7               print("false");
8       }
9
10      return 0;
11  }
```

9.17.109 test-if5.out

```
false
```

9.17.110 test-if5.tens

```
1   int main () {
2       if (0) {
3               print("true");
4       } else if (0 || 0) {
5               print("true");
6       } else if (0 && 0) {
7               print("true");
8       } else {
9               print("false");
10      }
11
12      return 0;
13  }
```

9.17.111 test-if6.out

```
success
success
```

9.17.112 test-if6.tens

```
1   int main () {
2       let a = 1;
3       let b = 4;
4
5       if (b == 4) if (a) print("success\n");
6
7       if (b == 4) {
8               if(a) {
9                       print("success\n");
10              }
11      }
12
13      return 0;
14  }
```

9.17.113 test-if7.out

```
success
```

9.17.114 test-if7.tens

```
int main () {
    % this else should stick onto the second if, so the program should print
    % "success"
    if (1) if (0) print("failed"); else print("success");

    return 0;
}
```

9.17.115 test-if8.out

```
success
success!
```

9.17.116 test-if8.tens

```
int main () {
    if (1) {
            if (1) print("success\n");

            if (0) print("failed\n");
            else print("success!\n");
    } else {
            print("nothing\n");
    }

    return 0;
}
```

9.17.117 test-indexpr1.out

```
1.000000
4.000000
```

9.17.118 test-indexpr1.tens

```
int main () {
    let T_{2, 4, 1, 6, 2};
```

```
    let W = [[1, 0], [0, 1]];
    let U = W_{a, b} + [[2, 3], [4, 5]];
    let a = 0;
    print(W_{a, a});
    print(U_{1, a});

    return 0;
}
```

9.17.119 test-int-ops.out

```
34
648
4
4
-25
25
19
1
0
1
0
0
1
0
0
1
1
1
0
0
1
0
1
1
0
0
1
1
0
0
1
```

9.17.120 test-int-ops.tens

```
int main () {
        print(12 + 36);
        print(43 - 9);
        print(12 * 54);
        print(23 / 5);
        print(20 / 5);
        print(-25);
```

```
        print(--25);
        print(+19);
        print(6 == 6);
        print(12 == 3);
        print(14 ~= 6);
        print(42 ~= 42);
        print(54 > 72);
        print(21 > 14);
        print(53 > 53);
        print(14 >= 52);
        print(85 >= 65);
        print(14 >= 14);
        print(76 < 91);
        print(12 < 4);
        print(84 < 84);
        print(17 <= 64);
        print(19 <= 3);
        print(28 <= 28);
        print(1 && 93);
        print(12 && 0);
        print(0 && 0);
        print(34 || 12);
        print(65 || 0);
        print(0 || 0);
        print(~6);
        print(~0);

        return 0;
}
```

9.17.121 test-loops1.out

```
15
4
3
2
1
5
4
3
2
1
5
4
3
2
1

1
1
1
```

9.17.122 test-loops1.tens

```
int main () {
	let i = 0;
	let j = 0;

	for (i = 0; i < 3; i = i + 1) {
		j = 5;
		while ( j > 0 ) {
			print(j);
			j = j - 1;
		}
	}

	print("\n");

	for (i = 10; i > 0; i = i - 1) {
		j = 2;

		while (j > 0) {
			print(1);
			i = i - 2;
			j = j - 1;
		}
	}

	return 0;
}
```

9.17.123 test-loops2.out

```
10.000000
1.000000
0.000000
2.000000
1.000000
3
10.000000
1.000000
0.000000
2.000000
1.000000
1
10.000000
1.000000
0.000000
2.000000
1.000000
0
10.000000
1.000000
0.000000
2.000000
1.000000
```

```
-10
```

9.17.124 test-loops2.tens

```
int main () {
	let i = 4;
	let j = 0;
	let T = [10, 1, 0, 2, 1];

	while (i >= 0) {
		for (j = 0; j < 5; j = j + 1) {
			% print all the elements of the tensor
			print(T_{j});
		}

		% decrement the counter in a variable way
		i = i - ftoi(T_{i});
		print(i);
	}

	return 0;
}
```

9.17.125 test-promotion.out

```
21.620000
```

9.17.126 test-promotion.tens

```
int main () {

	let a = 8;
	let b = 13.62;
	print(a + b);

	return 0;
}
```

9.17.127 test-scope1.out

```
4
5
3.141592
2.000000
2.000000
5
```

9.17.128 test-scope1.tens

```
int main(){
    let a = 4;
    {
        print(a);
        a = 5;
        print(a);
        let a = 3.141592;
        print(a);
        {
                a = 2;
                print(a);
        }
        print(a);
    }
    print(a);

    return 0;
}
```

9.17.129 test-slice.out

```
2.000000
32.000000
1079.600000, 1049.290000,
66150.400000
11103.152200
```

9.17.130 test-slice.tens

```
int main () {
    let A = [2., 34.34];
    let B = [[3., 32.], [2., 43.]];
    let C = [[[32., 23.],[32.3, 3.]],[[0.0, 23.33], [2., 323.33]]];
    let D = C_{a, b, c}*C_{c, d, e};
    let E = C_{a, b, c}*D_{c, d, e, f};


    print(A_{0});
    print(B_{0, 1});
    print(D_{0, a, 1, 0});
    print(E_{0, 1, 0, 0, 0}*2.0);
    print(C_{1, 3/3, 2-1}*A_{1});


    return 0;
}
```

9.17.131 test-sqrt.out

```
2.000000
6.000000
15.241391
```

9.17.132 test-sqrt.tens

```
int main () {
    print(sqrt(4.));
    print(sqrt(36.));
    print(sqrt(232.3));


    return 0;
}
```

9.17.133 test-stdlib1-pow.out

```
 4.000000
32.000000
125.000000
3262608926305880615636877866528931840.000000
```

9.17.134 test-stdlib1-pow.tens

```
int main () {
    print(pow(2., 2));
    print(pow(2., 5));
    print(pow(5., 3));
    print(pow(35., 23));

    return 0;
}
```

9.17.135 test-stdlib2-max.out

```
17.000000
3
```

9.17.136 test-stdlib2-max.tens

```
int main(){
    let T = [1,2, -3, 17, -20, 1];
    print(max(T));
    print(maxInd(T));

    return 0;
```

```
}
```

9.17.137 test-stdlib3-min.out

```
-20.000000
4
```

9.17.138 test-stdlib3-min.tens

```
int main(){
    let T = [1,2, -3, 17, -20, 1];
    print(min(T));
    print(minInd(T));

    return 0;
}
```

9.17.139 test-stdlib4-abs.out

```
0.000000
5.000000
5.000000
```

9.17.140 test-stdlib4-abs.tens

```
int main(){
    let x = 5.;
    let y = -5.;
    let z = -0.;

    print(abs(z));
    print(abs(y));
    print(abs(x));

    return 0;
}
```

9.17.141 test-string-ops.out

```
hello world!
hello world!
```

9.17.142 test-string-ops.tens

```
int main () {
        let h = "hello";

        print("hello " ^ "world!\n");
        print("hello " ^ "world" ^ "!\n");
        return 0;
}
```

9.17.143 test-strlen.out

```
5
8
2
```

9.17.144 test-strlen.tens

```
int main () {
    print(strlen("hello"));
    let s = "mystring";
    let f = "42";

    print(strlen(s));
    print(strlen(f));


    return 0;
}
```

9.17.145 test-tenexpr1.out

```
    1.200000
```

9.17.146 test-tenexpr1.tens

```
int main () {
    let a = 1;
    let b = 5;
    let c = 66.34;
    let d = 45+7-b;

    let T = [[a, b], [c, d]];
    let W = [[[itof(a)/itof(b)], [53*67]], [[43], [-41.4+b/d]]];

    print(T_{0, 0}+W_{0, 0, 0});

    return 0;
}
```

9.17.147 test-tensor-add.out

```
   2.000000
-646.000000
```

9.17.148 test-tensor-add.tens

```
int main () {
    let a = [[[1, 2],[3,23]],[[23.3, 4],[23., 2]],[[23., 42],[323, 2 ]],[[-323, 2],[3, 4]]];
    let c = a + a;
    print(c_{0, 0, 0});
    print(c_{3, 0, 0});
    return 0;
}
```

9.17.149 test-tensor-add2.out

```
6.000000
-1.300000
5.000000
12.000000
9.000000
```

9.17.150 test-tensor-add2.tens

```

int main () {
       let T = [1, 2, 3, 4, 5, 6, 7, 8, 9];
       let U = [5., -3.3, 2., -7, 1, 6, 1., 2, 0.0];

       let V = T + U;

       print(V_{0});
       print(V_{1});
       print(V_{2});
       print(V_{5});
       print(V_{8});

       return 0;
}
```

9.17.151 test-tensor-dotproduct.out

```
   47.000000
```

9.17.152 test-tensor-dotproduct.tens

```
int main () {
    let A = [2,3,4];
    let B = [4,5,6];

    let c = A_{i}*B_{i};
    print(c);

    return 0;
}
```

9.17.153 test-tensor-sub.out

```
    -1.000000
-1.500000
7.000000
```

9.17.154 test-tensor-sub.tens

```
    int main () {
        let T = [[1, 2], [3, 4], [5, 6]];
        let U = [[2., 3.2], [4.5, 2], [4.2, -1.]];

        let V = T - U;
        print(V_{0,0});
        print(V_{1,0});
        print(V_{2,1});

        return 0;
}
```

9.17.155 test-tensor-sub2.out

```
    5.300000
-3.000000
```

9.17.156 test-tensor-sub2.tens

```
    int main () {
    let a = [[[1, 2],[3,23]],[[23.3, 4],[23., 2]],[[23., 42],[323, 2 ]],[[-323, 2],[3, 4]]];
    let b = [[[-4.3, 2],[3,23]],[[23.3, 4],[23., 2]],[[23., 42],[323, 2 ]],[[-320, 2],[3, 4]]];
    let c = a - b;
    print(c_{0, 0, 0});
    print(c_{3, 0, 0});
    return 0;
}
```

9.17.157 test-tensoraccess.out

```
   1.000000
5.000000
```

9.17.158 test-tensoraccess.tens

```
    int main () {
        let T_{2,3};

        T_{0,1} = 1.0;
        T_{1,1} = 5.0;

        print(T_{0,1});
        print(T_{1,1});

        return 0;
}
```

9.17.159 test-tensorimmed.out

```
   1.000000
3.000000
10.000000
2.000000
3.000000
1.000000
6.000000
```

9.17.160 test-tensorimmed.tens

```
int main () {
        % 1-d tensor
        let T = [1.0, 2.0, 3.0];

        print(T_{0});
        print(T_{2});

        let U = [[1.0, 2.0], [3.0, 4.0]];
        let a = 1;

        U_{0,0} = 10.0;

        print(U_{0,0});
        print(U_{1-1,1});
        print(U_{a, 0});

        let V = [[[1.0, 2.0], [3.0, 4.0]], [[5.0, 6.0], [7.0, 8.0]]];

        print(V_{0,0,0});
```

```
        print(V_{1, 0, a});

        return 0;
}
```

9.17.161 test-tmult1.out

```
   74.680000, 1540.620000,
73.000000, 1472.000000,
92.000000, 1913.000000,

1251886.848000, 6993923.771598,
1786051.928400, 84575790.671598,

1219194.195200, 1768966.545858,
1307149.705980, 12485804.953858,


51273.740800, 6105572.018745,
572004.669372, 84111061.518545,

785685.388800, 84677919.251745,
8003784.826572, 1165796911.471545,



56732.800000, 99767.748170,
62321.768000, 822874.448170,

44807.008000, 2473898.081537,
254544.951500, 33843575.371537,
```

9.17.162 test-tmult1.tens

```
int main () {
    let A = [2., 34.34];
    let B = [[3., 32.], [2., 43.]];
    let C = [[[32., 23.],[32.3, 3.]],[[0.0, 23.33], [2., 323.33]]];
    let D = C_{a, b, c}*C_{c, d, e};
    let E = C_{a, b, c}*D_{c, d, e, f};


    print(A_{a}*B_{a, b});
    print(B_{a, b}*B_{b, c});
    print(E_{a, b, c, d, e}*A_{c});
    print(C_{a, b, c}*D_{b, c, d, e});


    return 0;

}
```

9.17.163 test-tmult2.out

```
   1760.000000, -252.677000, 979.700000,
106.336000, -121.571000, 111.992000,
968.547800, 23.111200, 19.566000,
103.189233, -19.755680, 139.566000,
6677.111200, 11.111110, 20.111200,

2360.600000, 5540.640000, 1204.400000,
1160.833333, -377.280000, 1262.000000,
10703.200000, -584.800120, 1984.000000,

13079.700000, 3401.840000, -1501.800000,
2294.793333, 2755.220000, -128.200000,
81536.600000, -2855.201107, -187.700000,


4963.840000, 3103.671200, 960.560000,
864.000000, 367.336000, 335.600000,
-195.200000, -2565.364000, 1285.610000,

1554.000000
1759.282000
```

9.17.164 test-tmult2.tens

```
int main () {
   let x = 34.4;
   let y = -1.4;
   let z = 42.0;
   let A = [1., 32., 23.];
   let B = [[0., .323, 3.], [32., 23., 4.], [32., -43., 8.2*4.5]];
   let C = [[[x, x*4.3, 7.2], [32., 23., 4.], [32., -43., 8.2*4.5]],
            [[y, x, z], [y/z, x*y, z], [x, x, x]],
            [[323., 4., 2.], [x, y, z], [2222., -.00003, 3.]]];


   print(A_{a}*B_{a, b});
   print(B_{0, a}*B_{a, b});
   print(B_{a, b}* C_{b, c, d});
   print(C_{0, a, b}*B_{b, c});


   let k = A_{a}*A_{a};
   print(k);
   let j = B_{a, b}*B_{b, a};
   print(j);

   return 0;
}
```

9.17.165 test-tmult3.out

int main() let x = 5.; let y = -5.; let z = -0.;
print(abs(z)); print(abs(y)); print(abs(x));
return 0;

9.17.166 test-tmult3.tens

```
int main() {
    let A = [2., 34.34];
    let B = [[3., 32.], [2., 43.]];
    let C = [[[32., 23.],[32.3, 3.]],[[0.0, 23.33], [2., 323.33]]];
    let D = C_{a, b, c}*C_{c, d, e};
    let E = C_{a, b, c}*D_{c, d, e, f};

    print(D_{a,  b, c, d}*E_{b, a, f, g,    c});
    print(E_{a,  b, c, d, e} * B_{a, b});
    print(C_{a, b, c}*D_{b, d, a, e});

    return 0;


}
```

9.17.167 test-while1.out

```
1
2
3
4
5
```

9.17.168 test-while1.tens

```
int main () {
       let i = 0;

       while (i ~= 5) {
              i = i + 1;
              print(i);
       }
       return 0;
}
```

9.17.169 test-while2.out

```
5
4
3
2
```

```
1
```

9.17.170 test-while2.tens

```
int main () {

    let i = 5;

    while(i){
        print(i);
        i = i - 1;
    }


    return 0;
}
```

10 References

References

[1] http://www.openicon.com/mu/math/tensors_01.html

[2] https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html#Real-Number-Constants

[3] https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html#String-Constants