Accelerator Final Project Report

Avi Jacob Chad-Friedman, Evan Charles O'Connor, Alan Phillip McNaney
{ajc2212, eco2116, apm2144}
12/22/2015

1.) Introduction

1.1) What is the Accelerator language?

Accelerator consists of a subset of the syntax of the R language. We will implement basic mathematical operators for vectors and matrices, boolean operators, if-elseif-else structures, for and while loops, and imperative functions. Our compiler will translate this subset of R to Accelerator enabled C++ to accelerate matrix mathematics and statistical analysis computation. Accelerator will implement a small standard library of functions, including commonly used statistical analysis functions from R such as mean, median, standard deviation into OpenMP. This will allow programmers and researchers familiar with R to write programs using known R syntax and still gain the performance improvements made possible via Accelerator's access to parallel computation resources. R is not typesafe. We intend to make Accelerator type safe.

1.2) Why R and OpenMP?

Large scale data is collected continuously from the internet and other sources by businesses, research organizations, and government agencies. This data can necessitate databases with records numbering into the hundreds of millions. This presents the very real challenge of efficiently and meaningfully interpreting that collected data. For instance, how do we sort it? CPU's are designed for general processing, and as such can carry out sorting and analysis on large scale data but are not specialized to this demanding and increasingly frequent task. What hardware resources are readily available and well suited to the task of large scale data manipulation?

Many hardware architectures already contain a large-scale parallel processing hardware device which is overlooked for the purpose of data analysis - the Graphics Processing Unit. The laptop on which this proposal is being written has 384 graphical shading cores clocked to 1029 MHz, and 128 ALU's in it's GPU alone.[1] For reference, this constitutes a mid-level Nvidia GPU. If we can leverage the parallel processing power already available within a system's GPU, we can make large scale matrix manipulation and the application of statistical methods to large scale data sets more efficient than would be possible with a traditional CPU.

Why did we choose to begin with a subset of R's syntax? R is a widely used statistical programming language, and over the last year has greatly increased in popularity. R has no native ability to access the parallel processing resources. Several major industry leaders, such as AMD, ARM, Cray, HP, IBM, Intel, NVIDIA, Oracle, and

[1] http://www.notebookcheck.net/NVIDIA-GeForce-840M.105681.0.html

others have been working since 2008 to create OpenMP - an API and language extension implemented in C, C++, and Fortran which allows access to the large scale parallel processing power of GPUs and other processing cores available to a specific architecture. Our intention is to allow access to the raw power of the GPU from the already familiar and easy to use context of R syntax, improving performance in matrix mathematics and statistical analysis of datasets by leveraging the parallel processing power available in OpenMP enabled C++.

1.3) Basic Syntax:

The syntax of Bar-R-CUDA is a subset of the syntax of R, and translates into CUDA enabled C to enable GPU optimization.

2.) Language Tutorial

2.1) Hello World

Accelerator, as mentioned above, contains a subset of the grammar of the R programming language. The specifics of that subset of grammar are discussed in the Language Reference Manual in Section 3. Immediately below is an example of the process used to turn an Accelerator program into a running executable, making use of a "Hello World" program written in the Accelerator subset of R:

HelloWorld.acc

```
print("Hello World")
```

Save the above line of code to HelloWorld.acc.

2.2) Creation and Use of the ./acc Compiler

You will need to be in a Linux / Unix / Mac command line environment. Unpack the downloaded tarball. From the base directory after unpacking our tarball, run the "make" command at the command line, as below. This will require the g++ compiler, version 4.2.1 or higher.

Command Line Input

```
$ make
```

Make will build an executable named ./acc which is the Accelerator compiler. Running the following command at command line will cause the ./acc compiler to ingest HelloWorld.acc and produce a semantically identical program in OpenMp enabled C++:

Command Line Input

```
$ ./acc < HelloWorld.acc > HelloWorld.cpp
```

Or, more generically:

Command Line Input

```
$ ./acc < [your_program_name].acc > [your_program_name].cpp
```

2.3) Compilation from C ++ to an Executable

Now that you have [your_program_name].cpp, passing this C++ file to the g++ compiler will produce an executable, using the following command:

Command Line Input

```
$ g++ -o HelloWorld HelloWorld.cpp
```

At this point, you will have a HelloWorld executable, and running as below will produce the below output:

Command Line Input

```
$ ./HelloWorld
Hello World
```

3.) Language Reference Manual

Below is an update of our earlier submitted Language Reference Manual. The introduction has been omitted as it is identical to the introduction above in section 1.

3.1) Data Types and Literals

- Note: all data types are immutable

3.1.1) Primitive Types

bool

- bool literal: a constant indicating true or false

- Examples of bool literals:
    - TRUE
    - FALSE

int

- int literal: an optionally signed string of digits with min/max values bounded that does not contain a “.”
- Examples of int literals:
    - 0
    - 1
    - -1
    - 123456789
    - -123456789

NA

- NA is identical functionally equivalent to R’s NA, which is a representation of value which is not available. It is essentially a null value. This can be assigned to variables or exist in any element in any matrix data type, and returns NA when evaluated using any operator against any other data type.
- Example of NA literal:
    - NA

double

- Double is a double precision 64 bit floating point type, as defined by the IEEE 754 standard with 1 bit for sign, 11 bits for an exponent part, and a 52 bit significand. Minimum and maximum values are identical to those of a 64 bit floating point type.
- examples of double literal
    - 123456.789
    - 0.0
    - -123.456789

3.1.2) Non-Primitive Types

vector

- A vector is an array type composed of primitives. They are an internal representation, and not available to the user outside of parameter passing inside of matrix creation. These are implemented using the C++ standard template library vector class.
- General Vector Creation Syntax
    - vectorIdentifier <- c(comma separated, same type literals or NAs)
    - whitespace ignored
    - Vectors cannot begin with an NA value.
- Examples of vector literals (literal bolded for emphasis)

    - intMatrix <- matrix(**c(1,2,3,4)**)
    - boolMatrix <- matrix(**c(TRUE,FALSE,TRUE,FALSE)**)

matrix

- Used to represent matrices. A matrix is internally handled as a vectors of vectors. Elements are represented by values stored via. An individual matrix may only contain elements of one primitive type, and may contain NA's. The type contained in a matrix is inferred at matrix assignment, and the type of a matrix cannot be changed once it has been created. Matrices cannot be declared without instantiation, cannot only contain NA types, and cannot have an NA in the first element of the first vector, and by extension from vectors above, cannot have NA in the first element of any row.
- General Array Creation Syntax
    - matrixIdentifier <- matrix(c(comma separated, same type literals or NAs), optional (nrow = int, ncol = int))
    - Whitespace is ignored.
- examples of matrix literals:
    - boolMatrix = matrix(c(TRUE,TRUE,FALSE,TRUE)) # array like matrix containing booleans
    - intMatrix = matrix(c(1,2,3,4,5)) # array like matrix containing ints
    - charMatrix = matrix(c('a','b','c')) # array like matrix containing chars, equivalent to a string type
    - naMatrix = matrix(c(1,NA,3,NA,5,NA)) # array like matrix containing ints and NA values
    - doubleMatrix = matrix(c(1.1,2.2,3.3,4.4,5.5)) # array like matrix containing double values
    - twoByTwoMatrix = matrix(c(1,2,3,4), nrow = 2, ncol = 2)
    - twoByWithNAMatrix = matrix(c(1,NA,3,NA), nrow = 2, ncol = 2)

string

- Equivalent to the R string data type, and is internally composed of char primitives in the style of R. Strings are delimited by double quote characters Allows for escaping meaningful characters. Double quotes are not escapable.
    - Escapable characters
        - '\n' # newline escaping
        - '\r' # carriage return escaping
        - '\t' # horizontal tab escaping
        - '\\' # backslash escaping
    - Examples of Literals
        - "my string"
        - "123456"
        - "\n"
- Character literals: A single char string containing an ASCII char.
    - Examples:
        - "a"

3.1.3) Casting

Casting between types is not permitted in Accelerator. The only exceptions to this are that each data type can be represented as a character data type for printing, and that any variable of any type may contain the value NA, with the above noted exclusion that vectors cannot have an NA in their first element and by extension matrices cannot have an NA element in the first element in any row.

3.2 Lexical Conventions

3.2.1 Identifiers

Identifiers refer to variables, functions, or formal arguments for parameter passing to functions. Identifiers are composed of alphanumeric characters, and must begin with a letter of the alphabet. Alphabetic characters can be upper or lower case.

3.2.2 Keywords

1. if
    - Used in if (boolean expression) { statements } and if (boolean expression) { statements } else { statements }
    - The first set of statements are only executed if the boolean expression evaluates to TRUE, and the second only if the boolean expression evaluates to FALSE.
2. else
    - Used in if (boolean expression) { statements } else { statements }
    - The second set of statements following else are executed should the boolean expression evaluate to FALSE.
3. function
    - Used in functionName <- function (formal params) { statements }
    - Indicates that the functionName identifier represents a function identifier.
4. for
    - Used in for (identifier in int range operator int) { statements }
    - Indicates a looping structure, causing the above statements to be executed repeatedly as many times as is indicated by the integer range statement
5. while
    - Used in while (boolean expression) { statements }
    - Break is not implemented
6. in
    - Used in for (identifier in int range operator int) { statements } looping structures
    - Associates an index variable for looping with a described range
7. TRUE
    - Boolean constant indicating true used in boolean expressions.

8. FALSE
    - Boolean constant indicating false used in boolean expressions.
9. NA
    - Indicates that a value is not available. Can be assigned to a variable of any data type, or as the value held in the element of any type of matrix.
10. nrow
    - Used as part of matrix creation in matrix(c(elements), nrow = <int>, ncol = <int>)
11. ncol
    - Used as part of matrix creation in matrix(c(elements), nrow = <int>, ncol = <int>)

3.2.3 Punctuation

1. ,
    - separates formal parameters in function definition
    - separates actual parameters in function calls
2. [ ]
    - matrix access
    - Examples
        - matrixName[a,b]
            - a and b represent valid indices, otherwise this will raise a compilation error
3. ()
    - expression evaluation precedence override
    - identifying boolean expressions for if - else statements
    - identifying iteration identifiers and ranges in for statements
4. {}
    - delineation of a scoped block of statements
5. “”
    - character data type declaration
    - Cannot be escaped as part of strings

3.2.4 Comments

- At present Accelerator does not allow for comments.

3.2.5 Operators

Arithmetic Operators

- Applicable to int, double, matrix (element wise, must match type)
    - + # addition
    - - # subtraction
    - * # multiplication
    - / # division
    - ^ # exponentiation
- Applicable to only integer
    - %% # modulus operator

Assignment Operator

- Applicable to all types
    - <- # assignment

Comparison Operators

- Applicable to int, double, boolean, character, matrix (element wise, must match type)
    - == # equality
    - != # non-equality
- Applicable to int, double, character, matrix (element wise, must match type)
    - > # greater than
    - >= # greater than or equal to
    - < # less than
    - <= # less than or equal to

Logical Operators

- Applicable to bool, bool matrices
    - || # OR
    - && # AND
    - ! # logical negation

Range Operator

- Applicable to int
    - : # range

Matrix / Matrix Operations

- Applicable to matrix
    - + # addition
    - - # subtraction
    - * # multiplication

Operator Limitations

- Expressions may contain Matrix / Matrix operations, or any other kind, but not both.
- Matrices must be of appropriately matching dimensionality in order for Matrix / Matrix operations to be used. Incorrectly dimensioned matrices will cause a compiler error.

Operator Precedence

- Operator precedence is the same as in C, both for regular operators and Matrix / Matrix operators.

3.2.6 Whitespace

- Includes
  - ' ' # space
  - "\t' # tab
- Does not include
  - '\n'
    - Newline characters are meaningful as statement separators

3.3 Syntax

3.3.1 Program Structure

Programs written in Accelerator consist of a series of function declarations followed by a series of statements, executed in order, which consist of expressions. Function definitions must come first, and mixing statements and function declarations will lead to compiler errors. In keeping with R's syntax, there is no enclosing function, simply function declarations followed by statements in a text file with the file extension ".acc". Accelerator is not object oriented, therefore there are no classes.

3.3.2 Expressions

Assignment Expressions

Assignment in Accelerator is carried out via the <- operator. It is right binding, taking a value from the right hand side and assigning it to the identifier on the left. As Accelerator is type safe, once a variable has been assigned a data type within a program, that variable must remain that data type. Attempts to assign data types other than that already set to a previously declared variable will cause compiler errors. In the case of assigning a function to an identifier, the block of statements that constitute the function body cannot end with an assign statement.

Example:

```
myVar <- “a string of characters”
myMatrix <- matrix(c(1,2,3,4), nrow=2,ncol=2)
myFunction <- function(i=0, j=0.0) {
        a <-
        print(a)
        return(a + 1)
```

Arithmetic Expressions

These operators represent basic mathematical operations, and are left associative.

Example:

```
5 * 5
25 / 3
1 + 2
2 - 1
10 %% 2
```

Matrix Arithmetic Expressions

These represent linear algebra operations on matrices. They must occur between matrices which are of the correct dimensionality for the matrix arithmetic operation.

Examples:

```
A <- matrix(c(1,2,3,4),nrow=2,ncol=2)
# Result:
#         A = [ 1, 3 ]
#             [ 2, 4 ]

B <- matrix(c(4,9,1,11),nrow=2,ncol=2)
# Result:
#         B = [ 4, 1  ]
#             [ 9, 11 ]
# Matrix addition
A + B
# Result:
#         [ 5,   4 ]
#         [ 11, 15 ]

# Matrix multiplication
A * B
# Result:
#         [ 31, 34 ]
#         [ 44, 46 ]

# Matrix subtraction
A - B
```

```
# Result:
#         [ -3, 2  ]
#         [ -7, -7 ]
```

Comparison Expressions

All comparison operators are binary operators comparing the left and right operands. They result in a bool. Comparison operators can only be applied to int and double.

Example:

```
1 < 2                    # bool (TRUE)
```

Boolean Expressions

Boolean expressions evaluate to bool data types and result in a bool. They can be applied to bool data types.

Example:

```
a <- TRUE
b <- FALSE
a && b          # evaluates FALSE
a || b          # evaluates TRUE
```

Literal Expressions

Any data type literal in the Accelerator language. They evaluate to their type and value as expected

Example:

```
1
1.1
true
“string”
```

Function Call Expressions

Expressions which call a previously defined function, causing execution of the function body, making use of the passed parameters and returning a value.

Example:

```
a <- foo(1, true)
```

Identifier Expressions

Expressions which are identifiers, and reference variables containing Accelerator data types.

Example:

```
a <- 12
a
```

The a on a line by itself above is an instance of an identifier expression by itself.

3.3.3 Statements

Statements in Accelerator are syntactically and semantically correct instructions composed of the expressions listed above, and are executed sequentially.

Expression Statements

Expression statements contain a single expression as defined in the expressions segment above and are terminated by a newline character.

If - Else Statements

Accelerator allows if-else control flow statements. A boolean expression is evaluated. If the boolean expression evaluates TRUE, the first block of statements is executed. If the boolean evaluates FALSE, the second block of statements is executed. Both the TRUE and FALSE blocks of statements must return the same type. The return type and value of a block of statements are defined by the value and type of the final expression executed

Examples:

```
if (boolean expression)
        { block of statements executed on TRUE evaluation }
else
        { block of statements executed on FALSE evaluation }
```

Looping Statements

Accelerator has two looping structures, a for loop and a while loop. The for loop uses an index variable which iterates across an indicated range, executing an attached block of statements as many times as is indicated by the range. The while loop evaluates a boolean expression. If that boolean expression evaluates to true, the attached block of statements are executed, and the boolean expression re-evaluated. If the boolean expression evaluates to false, the attached block of statements is not executed.

Example For Loop

```
for ( identifier in range expression )
        { statements }
```

Example While Loop

```
while (boolean expression)
        { statements }
```

Function Statements

Function statements invoke a previously defined function, causing flow of execution to transfer to the block of statements defined within the function and passing in any values passed as parameters. Functions make use of a “return()” sub-function to return values to their calling context.

Example Function

```
foo <- function(i=0){
        a <- i
        print(a)
        return(a)
}
```

Return Values

Each statement is evaluated to a value. Loops, and if-else statements return the value of the last statement evaluated within their associated block of statements. Both branches of an if-else statement must return the same type. Functions return in the fashion described in the function statement above.

3.3.4 Scope

Scope in Accelerator is divided between normal CPU memory space, and shared memory scope as defined by OpenMP enabled C++. These memory spaces are disjoint. It is possible to transfer data from the CPU scope to the GPU scope and vice versa. This is handled automatically by Accelerator, and is carried out when matrices are assigned and when legal arithmetic operations are performed on matrices. This implementation is intended to shield the programmer from being forced to manage disparate memory spaces, concurrent memory access, and synchronization issues. From the perspective of the programmer, there is one memory space.

3.3.4.1 GPU Memory Scope

When legal operations are carried out on two matrices, all information relevant to the operation is transferred to the GPU. These operations are then parallelized, results obtained, and transferred back into main CPU memory space. As such, operations involving matrices block until return values are obtained from the GPU.

3.4. Standard Library Functions

Accelerator provides one standard library function

Print

- print()
    - Writes the characters of vector to standard output
    - Can accept any Accelerator primitive data type, but vectors and matrices must be printed using a for loop or nested for loops.

3.5 Parallelizable Operations

User Experience

- Accelerator automatically parallelizes all matrix operations, including element wise operations. This limits flexibility, but completely abstracts the decision of whether a section of code is safe or wise to parallelize. The tradeoff heavily favors the programmer, who may be familiar with R syntax but not well versed, if at all, in multi-threaded programming. For small matrices, the overhead of sending data to and from the GPU is large; however, the vast majority of Accelerator use cases will involve large data sets and, therefore, large matrices. R itself already exists as a robust and efficient tool for the purpose of small matrix manipulation in a non-parallel computation environment. Automatically parallelizing all matrix operations covers a large subset of the cases in which multi-threading would be beneficial.

OpenMP Implementation

- Accelerator takes advantage of a core subset of compiler directives OpenMP provides. Again, the advantage of using relatively simple compiler directives is that the programmer can write useful parallelizable code without concerning his or herself with where or how the parallelization should occur.
- Matrix Operations
    - Matrix operations of the form A <operation> B, or A <operation> n where A and B are matrices and n is a scalar

-
```
#define CHUNK_SIZE 100;
int chunk = CHUNK_SIZE;
float** a, b, c; \\initialization of a and b
\\or int n; if A <operation> n
\\or #pragma omp parallel shared(a,n,c,chunk)

#pragma omp parallel shared(a,b,c,chunk)
{
#pragma omp for  schedule(static)
\\for loops to compute c
}
```
  - An operation within a loop structure of the form a = a <operation> b, where a and b are scalars
    -
```
int a, b;
\\some initialization
#pragma omp parallel for reduction(<operation> : a)
\\for loop with a = a <operation> b in the body
```

4.) Project Plan

4.1) Project Planning

4.1.1) Process for Planning

The Accelerator team met throughout the semester twice weekly, on Tuesday and Thursday evenings from 7:00pm to 9:00pm, and expanding later into the evening if necessary and more frequently throughout the week as was necessary toward the end of the project. At each meeting we discussed completed work in the days leading up to each meeting, problem areas which we were having difficulty bypassing, and whether we needed to seek outside help from our IA regarding persistent trouble spots in the coming days. We would work together for the allotted time and at the end assign work to be completed between the current meeting and the next scheduled meeting time.

Initial meetings focused on developing a concept for our language. We researched several different source programming languages and target programming languages and presented our findings to one another along with ideas for what about each source / target language pair would make for an interesting language in and of itself. Upon presentation of this research work, we quickly agreed and landed upon the idea of a language which made the parallelization advantages of GPU processing available in the increasingly popular context of the open source R programming language. We then researched different parallelization technologies, CUDA, OpenACC, OpenGL, and finally OpenMP which we decided upon as it was the only parallelization

technology which was compatible with the graphics hardware in every team member's laptop.

4.1.2) Process for Specification

The next phase of our project involved specifying exactly what subset of R's functionality we wanted to focus on. R provided a lot of redundant syntax. For instance, R will allow assignment via the "=" operator or via the "<-" operator. In each instance where we faced multiple choices for implementing language specifics, we decided on a single implementation. Via this process, we came down to a basic set of language features which would allow for us to implement at the very least an iterative version of a GCD algorithm and basic matrix mathematics including addition, subtraction, and multiplication of matrices which would be parallelization enabled via compiler directives from OpenMP. We specified this functionality in our Language Reference Manual, included earlier in this document.

4.1.3) Process for Development

Once we had identified which subset of R's functionality we wanted to implement, we began a regression test driven approach to development. We began with basics, getting integer literals and print statements working. Once we had print statements working, this allowed us to compile all the way through to executables and compare the output of our generated executables against expected output given a particular input program. We began with basic integer arithmetic operations. Once those were producing correct output, we progressed to the integration of variable assignment and boolean literals. Boolean literals allowed us to progress to boolean operators, in anticipation of control flow statements. In order to implement control flow statements, we needed to ability for our compiler to process multi-statement programs, so this was our next goal.

At this point we became capable of independently developing different portions of our language. We broke up language features into individual tasks and assigned them to teammates. In this fashion we implemented the floating point data type, printing of variables, if-else statements, vector data types, string data types, and matrix data types. As we approached the development of functions and other statement block structures, we realized that we were going to need to need a better implementation for the assignment and typing of variables. We restructured our compiler to incorporate an environment to track declared variables and their types. With this element in place, we finished developing all language features which would allow us to write a GCD function, and statement block structures which necessitated environment. Finally, we added matrix mathematics functionality and testing, along with benchmark testing.

4.1.4) Process for Testing

All testing was carried out via a regression test suite built inside of a shell script. The script compiles the ./acc compiler using a Makefile. Following successful creation of the ./acc compiler executable, the test script then iterates through all tests contained in the sourceFiles directory. Each .acc file in the sourceFiles directory tests a particular feature in the Accelerator language. Each .acc test file is ingested by the ./acc executable which generates a correspondingly named .cpp file in the compiledCpp directory. That .cpp file is then compiled with g++, and an executable with a corresponding name is created in the executables directory. That g++ generated executable is then run, and the output to standard out generated by running that executable is captured in a .txt file, correspondingly named, in the output folder. That output .txt file is then diff compared against an expected output .txt file in the expected folder. If the output text file is the same as the expected text file, the test is considered to have passed. If it is not, the test is considered to have failed. A single line is printed to stdout indicating whether each test has passed or failed. After running all regression tests in the suite, final results display the number of tests which failed, and the total number of tests.

4.2) Style Guide

- Given that the general idea of our language is to make parallel processing available to people who are already familiar with the R programming language, the style guide for our language is based on that of R, with a few minor departures. The below is adapted from Google's R Style Guide, and adjusted where appropriate. Rules are also omitted where they do not apply to Accelerator.
    - File names end in .acc
    - Variable identifiers begin with a lowercase letter, and are camelcase
    - Function identifiers begin with an uppercase letter and are camelcase
    - Line length - maximum 80 chars
    - Indentation - two spaces, no tabs
    - Curly Braces - first on same line, last on own line
    - else - surrounded with else braces
    - Assignment, use <- and not = (the = operator is not recognized by Accelerator)
    - Semicolons are to be avoided
    - Function definitions - keep parameter names and their default values on the same line, do not split lines

4.3) Project Timeline

Wednesday, September 30th	Formation of Group and Project Proposal
Thursday, October 22nd	Scanner and Parser Complete
Friday, October 23rd	First Code Generated
Monday, October 26th	Language Reference Manual and Initial Grammar for Accelerator Complete
Monday, November 16th	Initial Regression Test Suite Formed and Initial Hello World Complete
Tuesday, December 22nd	Full Compiler Structure Complete
Tuesday, December 22nd	Project Presentation and Submission of Final Project Report

4.4) Roles and Responsibilities

We were a group of three, due to a member of our initial group dropping the class. As such ,there was crossover in the roles carried out by each member as workload intended for four students was distributed. All members of the team contributed to all portions of the compiler.

Avi Jacob Chad-Friedman	Compiler Back End, Front End, Code Generation, Automated Testing Suite, Semantic Checking, Symbol Table and Environment Handling
Evan Charles O'Connor	Compiler Back End, Front End, Code Generation, Automated Testing Suite, Semantic Checking, GPU Functionality Testing
Alan Philip McNaney	Compiler Back End, Front End, Code Generation, Automated Testing Suite, Project Management

4.4) Software Development Environment

4.4.1 Tools

- GitHub Repository - used for version control and as a code repository.
- G++ - used for compiling our C++ code to machine executables
- Slack - used for project communication
- Vim - used as an editor for Ocaml, C++
- SublimeText 2 - editor for Ocaml, C++

4.4.2 Languages

- Ocaml 4.01.0 - for scanning, parsing, building symbol table, semantic checking ,translation of R types to C types, and C++ code generation
- C++ 4.8.4 / 4.2 - Target language we compiled into from Accelerator
- OpenMP - used for adding compiler directive pragmas which allowed parallelization of matrix operations in C++

4.1) Project Log

Below are the entries on the Master branch of our Git Log.

commit 3ff155537cec7c8ca543a5db35b12d1ac25372e8
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 14:36:11 2015 -0500

Added some features for openmp

commit c2b586fbab035275040a1ab903e1a382515a400a
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 14:22:58 2015 -0500

Compiler directives working and hello world
test

commit 22e60c3e3910e62ffcd081f846414014c1c938c1

Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 13:53:25 2015 -0500

    More last minute clean up

commit 09cbbfafd5ffaac3f0aed2bbdcb3aa428abf0cba
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 13:39:20 2015 -0500

    sast clean

commit 0ce10ffabd30018068bf13dc5c7afc1be05586e7
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 13:29:44 2015 -0500

    more cleanup

commit aa4d5b517b589cf7653cfc48721db5cb502d72b4
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 13:21:21 2015 -0500

    Tidying up

commit 640d558249854c89f94a10257c2ac9af6b898c54
Merge: 4a984cd d476762
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 13:10:37 2015 -0500

    Merge branch 'master' of
https://github.com/achad4/Accelerator

commit 4a984cdaf47692eba29b0882de558030b48605c3
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 13:10:01 2015 -0500

Matrix mult tests

commit d47676201ef638caefb65c6b2481a1009d4693fb
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 22 13:03:19 2015 -0500

yoooyoyoyoyoyoyoyyoyoyoyoyoy

commit 7af33575c112b09c2aee97fa858ae956d43513ef
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 22 12:01:08 2015 -0500

all tests pass.. for real

commit 7347fce967270814c0ae439f6596dc2d5b57ffce
Merge: 80e858f 4501910
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 11:44:19 2015 -0500

Merge branch 'master' of
https://github.com/achad4/Accelerator

commit 4501910c00fd5cc564f379d90e944e76b36dcf74
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 22 11:42:20 2015 -0500

everything but matrixTest

commit 80e858f75375a9f531fb52e9c3a3c99729df8cfe
Merge: 52407b3 5163046
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 10:42:52 2015 -0500

Merge branch 'master' of
https://github.com/achad4/Accelerator

commit 52407b3e423a6315551e2ab9acf3dc1ea015c879
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 10:42:47 2015 -0500

    Cleaning up benchmarks

commit 516304654fc7b2764ad88080068c1ba8dc37c665
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 22 10:40:46 2015 -0500

    all tests pass

commit 3d553af9fc9063ac40c0ddcfb7f4b62e6521bda5
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 10:34:03 2015 -0500

    Basic python test working

commit 5bc3b1855c13f13578d720a59c723c49002805c1
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 22 10:09:14 2015 -0500

    new tests

commit 332dbfd98326d54c13c84534041dd84bec939c24
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 22 10:08:26 2015 -0500

    matrix mult passes

commit 7b67155ed5fcdfa56f025708e1d6f54bce5d2f57
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 22 09:58:16 2015 -0500

c++ work

commit 0185862d79c76a089406d7a306825287143ac753
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 22 07:59:04 2015 -0500

working-kinda

commit e5588761994644ef2ddfeab41f8aed9cbfaf8499
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 06:25:51 2015 -0500

Fixed misc bugs

commit a703746ccc984a5a112822641214571a77a332fa
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 05:50:34 2015 -0500

new c start

commit 6fd9b70d34e6d2e02c7f43358ab84205e4229c5a
Merge: c8038b9 52fc5c1
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 05:45:36 2015 -0500

Merge branch 'matrix_ops' of
https://github.com/achad4/Accelerator

commit 52fc5c1e2c5a6c0a769bfe51a8d2f0f605ec1010
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 05:29:36 2015 -0500

Remove logs

commit 0a7fd6d86d3903b1d1a9eaebdf10bdd185ccbac9
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 22 05:28:30 2015 -0500

    matrix hacks

commit 8f5e6db474f8b997001233f61cd9ca675a183a16
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 22 04:44:53 2015 -0500

    c++ code

commit 9477608ca3863b78116225fe54d0b3567d9458c6
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 22 03:23:30 2015 -0500

    hack for matrix

commit 5565a59a1af71ea85200745a783bfb6364198043
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 22 02:42:03 2015 -0500

    matrix add hacking

commit c8038b99f43a3803cc9d0f7f308312a9d0cb5fe4
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 02:41:29 2015 -0500

    Fixed scoping and block edge cases and tests

commit 4f0f766ba6247647e67984e7f4aac6b6440f5cc4
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 22 02:23:43 2015 -0500

c++ stuff

commit 1bfe749d8b79f9ff445b2eff620f397c93460705
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 02:01:44 2015 -0500

Think I fixed blocks

commit bbed2a07dfcaa720f488ce5b9e099b86fa74357a
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 22 01:34:59 2015 -0500

tests

commit 12b06f14c7e1125603feb2125aa295ceec88d971
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Tue Dec 22 01:22:13 2015 -0500

odd behavior in compiling matrix assign and
matrix add - print statements w/ newlines added for clarity

commit d17752e89c2df98bf6ab2b86fd0807da08e233e7
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 22 01:03:28 2015 -0500

environments motherfucker

commit 4906d5a725ac562ba84abaa630fc28fb7d828e51
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 00:38:43 2015 -0500

fixed brackets

commit ff9162abcba065ce4eb6001aeb7d7c93835f1898
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 00:26:21 2015 -0500

    Reverting

commit e8d947da2578387d5455773d4dd6761ae173184d
Author: evanoconnor <eco2116@columbia.edu>
Date: Tue Dec 22 00:08:13 2015 -0500

    log statements for debugging

commit ff1f28ad526233a038df55d8fe177e4f8ef7ae09
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Mon Dec 21 23:15:12 2015 -0500

    matrix fix

commit f7a0a1e175cc82f6fa966b3526c968ccd9806123
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Mon Dec 21 23:05:47 2015 -0500

    matrix add work

commit b803a0d137140ee149c0197f9ebf9f97503f405a
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Mon Dec 21 22:34:30 2015 -0500

    added matrixAdd.acc

commit a18f72e3f96fa19ff7070da2c7c6ccff8bf6d329
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 21:40:10 2015 -0500

gcd test works

commit abb3c3784b30d4be39474ae687755ae7af0ff0f9
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 21:27:17 2015 -0500

While test working

commit 89b872f570553fcec91561cb086912ac915e72c0
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 21:24:55 2015 -0500

Can't cast types

commit 85a79b752f73ffea4b55bd1cca2d853d67f386a3
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Mon Dec 21 21:19:29 2015 -0500

not getting errors from our compiler, but
still compiling out c <- a + b instead of actual matrix building

commit 5e5e7cc0b2545479dc1a8a787e612628e4ed1004
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 21:19:22 2015 -0500

Function test working

commit 8e80ff7769d11cdb4ac7275b939cde2a0b0a6fcd
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 20:53:54 2015 -0500

Assign / update bug maybe fixed

commit fca57abba8a3f400aab7a6bd9e8783b72ecedb6c
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 20:46:10 2015 -0500

    fix

commit 5467a67decd0a6612760221530719bcbeb3960db
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Mon Dec 21 20:29:36 2015 -0500

env

    still trying to get matrix addition up and
running - it still compiles c <- a + b despite my best efforts

commit 1c55139e6fbdb403ce6bf47c834e57539c03ddaf
Merge: 4f77b1b 8bd929c
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 20:27:00 2015 -0500

    Merge branch 'evano' of
https://github.com/achad4/Accelerator into function_work

commit 4f77b1baa67bf02f1de8dc3a3c3b4d8f20822046
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Mon Dec 21 19:43:03 2015 -0500

    bug fix

commit cb73ff6b12fbc16deca35d3b2137de463da32938
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Mon Dec 21 19:39:49 2015 -0500

    local scopes for environments

commit 8bd929c51b7e0909fd1a3ec511a1432f60ac0cd1

Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 19:28:51 2015 -0500

    flip flop env bug

commit b579e97cd95a6880a0417a0f242adba383639e78
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Mon Dec 21 18:54:52 2015 -0500

    environment.ml

commit 552ed59741049379077341eb0b512afef9b849a3
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 18:36:10 2015 -0500

    update work

commit 46141e913e69bf17ceaeb9694b06e4312c73d0e3
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Mon Dec 21 18:22:16 2015 -0500

    partial work on matrix addition - don't know
how to pass result matrix to c, as technically they're different
operations

commit a4f94ffaa67d769566b2e4b64732d54a5ca0b72d
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 16:20:16 2015 -0500

    while tests

commit 72fec2534c381cf8f4dcc261c93e325d6f57704d
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 16:19:59 2015 -0500

More while work

commit 28d8f5f6e0ca12e8847c234611f68c32b13b2874
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 15:59:48 2015 -0500

While loops basically working

commit 29d59ef512c2d34f15e069d62c88e64152f9626a
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 15:49:20 2015 -0500

while through env

commit 8670bbc0461f209a3223f53849b71cec6b86acc4
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 15:24:00 2015 -0500

include string.h

commit 4b583a6fc4e6e3f328032149f1bc04af2c050516
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 15:18:07 2015 -0500

Fixed string compare bug

commit 3f695d877acd93578e1a58ce47eb26f995691a0c
Merge: 4b69edd 3c2c2f3
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 15:09:47 2015 -0500

Merge branch 'eq_neq' of
https://github.com/achad4/Accelerator into evan_eq

commit 4b69edd6bbd1d37b265c76903c8ca3283f94e8f3
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 15:09:21 2015 -0500

    Fixed func1 test

commit 3c2c2f3baf4b91f67203ed171fa9c2aeed37151d
Merge: e9c0e81 b3bd5f8
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Mon Dec 21 15:04:20 2015 -0500

    Merge branch 'eq_neq' of
https://github.com/achad4/Accelerator into eq_neq

commit e9c0e8144ebe622bb37965ad0104d05d9791a854
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Mon Dec 21 15:04:00 2015 -0500

    added separate method of comparing strings
to deal with older compilers

commit 45c32b316e745ab60f4b3207e368f57feeaf7588
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 14:47:05 2015 -0500

    Fixed backwards ordering of blocks

commit b3bd5f8b646b1631b8335747df03d22c0177a0cd
Merge: 6ef5664 37ffb52
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 14:38:12 2015 -0500

    Merge branch 'eq_neq' of
https://github.com/achad4/Accelerator into eq_neq

commit 37ffb525983b3be0195b63f92e9d7fe000f7bf52
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Mon Dec 21 14:33:57 2015 -0500

    == and != working once again

commit 7ddafab7a1beb287970ce832a29d0213befcfb3b
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Mon Dec 21 14:27:06 2015 -0500

    compiling all the way to ./acc now with ==
and !=

commit c8c2d0a5acbb99a3d0bed3ea7338129ed94d10a3
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Mon Dec 21 14:14:07 2015 -0500

    added eq neq to scanner and parser

commit 6ef56646dd63f5c818d8c828e95a9269ccd31c18
Merge: 4c9c421 d7dc1ea
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 14:02:52 2015 -0500

    Merge branch 'new_functions' of
https://github.com/achad4/Accelerator into eq_neq

commit 4c9c4214db7094907eff284af9313d00ca1643ed
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Mon Dec 21 13:19:01 2015 -0500

    got == and != working for int, float, bool,
string

commit d7dc1eace763a7c91c148725d10b21d656c9d4d4
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Mon Dec 21 13:15:33 2015 -0500

    fixed stmt ordering in blocks

commit 3c2836ef7a47fb602c666e89b412e3e51c920c7e
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 12:41:01 2015 -0500

    trying to fix envr bug

commit 13298c8d6ae381c66fe7ce005028685ad69d7e49
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 10:45:14 2015 -0500

    Function block bug work

commit e5841cfe610aa8f7e1a6cc10e7beee59acd0810f
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 09:57:23 2015 -0500

    functions working again

commit f57a16265eca80b417d31ec83b1ca102a4aca20e
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 09:44:11 2015 -0500

    All tests except func1 working

commit c27302f92d0fd63dc1ccc4a84ff3680c7f073655
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 09:28:05 2015 -0500

Done with functions for the moment

commit 35aeaf61689f677b387bc96521af6db807784773
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 09:16:17 2015 -0500

Assigning variables to functions with type

commit c584e4f78040471242c06e869b22310b74bec997
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 08:55:39 2015 -0500

Added back in user defined function calls

commit 90f4480f5adc8c77b13e012e97068f6645e844ca
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 08:51:52 2015 -0500

Print working again

commit ecf4429dee9dc69038c760c2cfa321c77e1b7590
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 08:35:25 2015 -0500

Added print function def

commit 702753fe9fd6c0dc8ddf48b1cd75ddd85ef48c29
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 02:58:03 2015 -0500

Formal defaults working

commit f39edc2baa34636ca84340d70d6147896637298b

Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Mon Dec 21 02:50:03 2015 -0500

    match fix

commit 1814b9a8dc19f49968c7b943e4188a2ee9ddcb88
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 02:28:24 2015 -0500

    Assignment / envr bugs fixed for the moment

commit ebf8b495450ccab60d5ebc107fceca077ab8edd8
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 02:22:52 2015 -0500

    Something is working maybe

commit 073595bd7d7add1b18afe768bf26ad871dfa144a
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Dec 21 01:42:51 2015 -0500

    Hack pt 1

commit b75227bd6b4657e549c1e62787c89d5188e007dc
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 23:11:08 2015 -0500

    prints

commit 4ee4e1d066852e70b2a24900d38626ce71f4bf5d
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 22:42:40 2015 -0500

    finding assignment bug

commit b13d14ec6b8bf5b4a2740e9553afd6558f59ed61
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 22:09:09 2015 -0500

    environment work

commit c4b6ddd3f7bdfd018e8a473b306bc12a8408c838
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 21:24:00 2015 -0500

    environment tweak

commit 6b8c805a4a28562d3875388fb43358fd3a592aa4
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 19:39:52 2015 -0500

    new test

commit c847985ab4bbd06751b0972e6c2be03b067edd90
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 19:20:55 2015 -0500

    attempted matrix fix

commit 08490d2de6315c885949ef3eab7c948bd909b038
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 18:42:15 2015 -0500

    funciton tests

commit 8d17c1c34b993f4c29a0dde8cd6d3a0466f2b225
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 16:13:07 2015 -0500

bug fix in add_return

commit 879be58bb3f1fa46e432b26d191b1c51a6641a0d
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 15:05:56 2015 -0500

adding return types to functions

commit e3e380c7b4f3646656f9538a23800a12df4c1ef8
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 14:06:38 2015 -0500

cleaned code

commit ae8491badf59c155b787160fc66ed730331f57ea
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 14:00:16 2015 -0500

took out extraneous test

commit 64327d3231bb6d2d3bd7cd31cd17aa640b195f78
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 13:55:25 2015 -0500

fixed vector/matrix parsing

commit 5f442202759907f178c0103a64f244341f0110d5
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Dec 20 13:31:08 2015 -0500

fixes statement reversal problem in cast.ml
by reversing the list of statements comprising the body of main

commit 207d79ab439f1aa5c1bb32b3b5c17963e0afe2ce
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 13:05:08 2015 -0500

    grammar fix

commit d5b5bdfada0c6b77862852afba46b87a476ec37c
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 12:48:41 2015 -0500

    scoping functions

commit b628aa573362a8e1528c7594e36f8b2f9c0a8d0a
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 12:20:48 2015 -0500

    restrucutred program for better functions

commit 473432735fdf0685607c95386a21bd9528b29705
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 11:27:07 2015 -0500

    types of blocks and fcuntions from last stmt

commit 548e4eac478bf74370983b8ffe4aebefc93c2be9
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 11:02:27 2015 -0500

    no longer require return stmts fom function

commit 5360aa5ba2e1eaa355183107229c920c2c1b00b7
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Dec 20 10:19:35 2015 -0500

changed grammar

commit cdeb55af4b976dd3fa68a4acf7e9decd31db43b7
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Dec 20 07:39:47 2015 -0500

Function arguments / return type appear to
be working

commit fcf019c4bb8e5e347c94ec61717c6ef775582e41
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Dec 20 06:41:52 2015 -0500

Function declaration compiling all the way
through

commit fa97d0e8b40052c0324219c75748d50a22b49dc2
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Dec 20 06:31:27 2015 -0500

Must give formal params default valeus up to
cast

commit 38c1ea5828c589ff1a01215322f5fda003e93201
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Dec 20 05:30:06 2015 -0500

Func decl up through cast

commit bf0d934243b9fc614ff6e084a8ab82c5a574c4ab
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Dec 20 03:20:44 2015 -0500

Fixed matrix test

commit 739076ff0f3d966ec40429783726e8084cbf2af3
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Dec 20 03:04:23 2015 -0500

    Fixed vector matrix type problem

commit b07741088f2ad37449c3494615892539c47d347b
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sat Dec 19 23:25:48 2015 -0500

    having trouble with types, matrix, vector

commit 75b7d0d4d05baff45012106b57e86950560e18ac
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sat Dec 19 20:03:31 2015 -0500

    pattern matching and bug fix

commit f561d9bd7587107245c898d1c464a3bdb4e5e74c
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sat Dec 19 19:48:14 2015 -0500

    fixed pattern matching for type_match in
environment.ml

commit 9f7d72666260cb1edad91af1dda3a4955e3f80f4
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Fri Dec 18 18:51:28 2015 -0500

    fixed type match problem with arithmetic
operators. Vector and matrix tests only tests failing now.

commit f8017e98cf33dc9663c13daf69402f08a33a4f4b
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Dec 18 18:06:09 2015 -0500

    deleted print

commit 8d75da727fa2d17b65112531375b956bb630fded
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Dec 18 17:58:19 2015 -0500

    environments for allmake cleanmake clean

commit d48be5a5c629147d35386fcfae57a82fb9a8c8b9
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Dec 18 17:55:14 2015 -0500

    cast compiles again

commit 30b39346adf2982c62a19df2c89fe8a3945c107d
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Dec 18 17:46:23 2015 -0500

    cast work

commit baffc709845f9f7fba12fc095b22e3f7f53b49c7
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Dec 18 17:38:16 2015 -0500

    sast compiles again

commit a3163c051ac8e29c92c409774cd8a0cb680ddd95
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Dec 18 17:37:43 2015 -0500

    sast compiles again

commit 743fa58956a6a802683462563a35cc5ac6f1d43d
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Dec 18 16:36:40 2015 -0500

    environment compiles again

commit dc732200453739fb3449df3c99fde83d3a6a31fc
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Dec 18 16:14:10 2015 -0500

    refactoring environment

commit 878de0f5607456bb1b74ba630c81913165eafb0b
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Dec 18 14:55:58 2015 -0500

    environment refactoring

commit d19b64d8c8af6f0f9fbe8f8231ab5e648f5ef560
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Dec 18 14:21:28 2015 -0500

    merging

commit 8682aaa8f4dbe9e7b5abf3dd4afd32254805bd0d
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Dec 18 14:03:44 2015 -0500

    merge issues

commit 363f879adecf010a661cb0990bc03a46fd9c2488
Merge: 4217aaf e56fb7c
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Dec 18 13:54:14 2015 -0500

merged stack

commit 4217aaf3561f9d78c6e758138c27a12fb352c7b8
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Dec 18 11:40:42 2015 -0500

shiiiiiit

commit e56fb7c472b593929bb855bcb53f66ecfc9d0736
Author: evanoconnor <eco2116@columbia.edu>
Date: Fri Dec 18 11:16:19 2015 -0500

All tests pass

commit dcf0cf682027cb9d1674b9451a3d1166ba6c9481
Author: evanoconnor <eco2116@columbia.edu>
Date: Fri Dec 18 10:57:24 2015 -0500

Basic int assigning seems to be working

commit 0c68a0281c6f7c3ac3f6048f94aac154d4cf6fbe
Author: evanoconnor <eco2116@columbia.edu>
Date: Fri Dec 18 10:47:57 2015 -0500

Maybe working in reverse order...not sure

commit 0e677e001027767499c7550a52c6a1ca9c7ba0a3
Author: evanoconnor <eco2116@columbia.edu>
Date: Fri Dec 18 08:13:45 2015 -0500

Compiles all the way through with new
environment.ml - yet to test

commit 82f007e02ce267224f60ff9fc963ffd110db72c0
Author: evanoconnor <eco2116@columbia.edu>
Date: Fri Dec 18 08:09:59 2015 -0500

    Compiling up through acc no idea if it works

commit 9c8ba997f2aecfe729aa44c921968e715586aa9b
Author: evanoconnor <eco2116@columbia.edu>
Date: Fri Dec 18 05:58:11 2015 -0500

    SAST work mostly - moving enviornemnt after
sast probably now

commit ec98ab6f481b72b772b8adb97305292990fbcdae
Author: evanoconnor <eco2116@columbia.edu>
Date: Fri Dec 18 04:19:19 2015 -0500

    Early environment stages

commit 7d5d98191af08cc2e30d18a2c07e67ff027812b1
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Wed Dec 16 17:53:15 2015 -0500

    cant get variables working yet

commit 71e4aebf61f164223e3971ba59c58a82801d48e7
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Wed Dec 16 16:15:38 2015 -0500

    more work on symbol lookup

commit d0591a49dbed427b7df113b633a776bfcbca2e02
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Wed Dec 16 15:29:00 2015 -0500

symbol table work

commit 33b2db152061b490496290174549a584d2d84713
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Wed Dec 16 15:27:37 2015 -0500

symbol table work

commit 410541e3d422cf5414d15f11fb925c9cd7f5d2fc
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Dec 13 04:33:39 2015 -0500

Fixed warning about matrix id access

commit d775db29f8296faeaf2353558f039730c0ac8d06
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Dec 13 04:30:09 2015 -0500

Can now specify one test to run

commit f494a38822636b9ba1da0102638c1cb724842392
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Dec 13 04:25:15 2015 -0500

Cleaned up test suite

commit a88998849aac20e70f407e00d9655b87bb820f3f
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Dec 13 04:13:55 2015 -0500

Added strings and fixed negatives

commit 606292712f9c0a82fac209040de249046db96a7b

Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Dec 13 02:36:38 2015 -0500

    Added negative numbers

commit 51d4a17d3458154b801f44ec4108557e16f83d93
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Dec 13 02:34:02 2015 -0500

    Got matrices working

commit ca3507df7d9f77b70d2838c817b2a7e0ff35c234
Author: evanoconnor <eco2116@columbia.edu>
Date: Sat Dec 12 23:54:23 2015 -0500

    Matrix access building with a warning

commit c2fafd0ed1d00e95db6181a09010d1bf2b8b2f92
Author: evanoconnor <eco2116@columbia.edu>
Date: Sat Dec 12 23:36:44 2015 -0500

    Matrix assignment working

commit 21d367bbbf7241fefb3678e8584637261524fb07
Author: evanoconnor <eco2116@columbia.edu>
Date: Sat Dec 12 21:53:01 2015 -0500

    Working on matrices

commit b71ef55ef0768f3c0483e88c4e458621ed95fc69
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Fri Dec 11 21:36:59 2015 -0500

finished making array tests fully functional
by adding for loops and array access functionality

commit 03dd79e7163df8e1e42326e4e6b80f441677bc4b
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Fri Dec 11 17:11:35 2015 -0500

working for loops

commit 7ed7e4504dfd8a820cea4d2efed9e1f548801ac3
Merge: e75b90c 156577c
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Fri Dec 11 16:37:23 2015 -0500

Merge branch 'master' of
https://github.com/achad4/Accelerator into forloop

Conflicts:
test-lib/compiledCpp/addTest.cpp
test-lib/compiledCpp/multTest.cpp
test-lib/compiledCpp/subTest.cpp
test-lib/parser.mly
test-lib/sast.ml

Merging master with cleanup into forloop.

commit 156577cdd1d2791c26bb11d31dedc1e06261b420
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Fri Dec 11 16:35:40 2015 -0500

removing .cpp files that have been tracked
and shouldn't be

commit df22dd70c486b8a67924a3717051e6ef93e369b4
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Fri Dec 11 16:30:05 2015 -0500

got vectors working for bools and floats as
well, test still in, currently failing due to lack of for loop

commit e75b90ce2090bf2232ebba2d166d5c6f6952b4c0
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Dec 10 21:07:32 2015 -0500

for test

commit 7065513eadf5fcb9acb3a934ca830f9f61efc48e
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Dec 10 20:58:47 2015 -0500

initial for loop work

commit a6249772dc93c1c1a1108a4d625bb747f483c4cd
Merge: ceaab63 fe61453
Author: evanoconnor <eco2116@columbia.edu>
Date: Thu Dec 10 20:26:21 2015 -0500

Merge branch 'master' of
https://github.com/achad4/Accelerator into ifelse

commit ceaab637ed40a941298ebdec6c83105c753b9ead
Author: evanoconnor <eco2116@columbia.edu>
Date: Thu Dec 10 20:20:53 2015 -0500

ifElse test working

commit 8ad51011415504178a936454df1f66a58d4793f4
Author: evanoconnor <eco2116@columbia.edu>
Date: Thu Dec 10 20:11:18 2015 -0500

Got if else working nicely

commit fe61453e096ec8ed11891a5de36324884cfa04a7
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Thu Dec 10 19:26:17 2015 -0500

    got vectors working! need to work on for
loops to print them

commit 52d705f807a3b9ab2bad8042fc1ca7d89cb9d1d5
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Thu Dec 10 15:18:31 2015 -0500

    got vectors working through to acc.ml,
trying to figure out how to compile as c++ requires a name

commit ee1d1ef5e84c6ee4000af5d85809a778ec815513
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Tue Dec 8 21:35:13 2015 -0500

    pickup with cast.ml int_vector
implementation

commit 06d668911e344c93c35df3636aec2f6d0da12e06
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Fri Dec 4 20:07:54 2015 -0500

    added ifElseTest.acc to sourceFiles in
ifelse branch

commit bc9ffcbf16444546548ff05b701b1fb1928c373d
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Fri Dec 4 19:46:07 2015 -0500

made master clean again by commenting out
ifelse stuff

commit b0e5f73bcdf5f2aedde60a43bc400ccded8e9ddf
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Dec 3 20:52:23 2015 -0500

ifelse

commit 2c1e838f19ff6d11d8b3b63fb7350b1e43c1bc58
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Dec 3 20:05:10 2015 -0500

ifelse

commit 804c708ad12fd31532c124a0db74c44603602777
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Dec 3 19:08:57 2015 -0500

Fixed tests with CAST

commit 9d0cc782eef77677f9608421233cfe5b9153fef4
Merge: 0014d90 51c3b32
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Dec 3 18:58:17 2015 -0500

Merge branch 'master' of
https://github.com/achad4/Accelerator into cast

commit 0014d908885711b9f80c2e55e6926ab3b993d704
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Wed Dec 2 15:38:58 2015 -0500

got rid of non-exhaustive matching error in
CAST

commit b3aa2aabee29faea23c8bcaab05f4c57070716cc
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Wed Dec 2 15:31:33 2015 -0500

    bug fix

commit a4578c0a17bd3d45ac3e8a6e33afee90631b2a12
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Wed Dec 2 15:26:35 2015 -0500

    fixed CAST shit

commit af1069c409edde8f83470319ac7769d35db78e49
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Wed Dec 2 15:08:21 2015 -0500

    got rid of type errors in CAST

commit 5a946e68aad6f265eb51f3fc26977c44995c3177
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 1 20:16:38 2015 -0500

    CASTT

commit 51c3b32d0fab53e597da922de13ae2ee9999139f
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Tue Dec 1 13:34:58 2015 -0500

    changed arith_expr and num_data to int_expr
and int_data, more in keeping with our data types in terms of naming

commit ce86e0c04dd611198808cebdb237a94a63890bf2
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 1 13:11:18 2015 -0500

...

commit 17f134445f2cda0a246c6cdd2543ec198f1e97d9
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Dec 1 00:14:57 2015 -0500

more CAST shit

commit 2e55b64ad937b31b77e5d3022588cebe95912728
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Mon Nov 30 23:38:54 2015 -0500

Fixed assign and print

commit d76e64ecb8f87c9f25bf703f938e4123c14533f3
Merge: 93bde6b a2e5dbf
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Mon Nov 30 23:17:55 2015 -0500

Merge branch 'master' of
github.com:achad4/Accelerator

commit 87759de818cf637e137126d13de567539ca7fdda
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Mon Nov 30 22:38:15 2015 -0500

Started CAST

commit a2e5dbf4e3037a96c714e2ed882c41c5e112808f
Merge: c605dc9 9fd4e96
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Mon Nov 30 19:52:57 2015 -0500

Merge branch 'floats'

Merging the floats branch changes into
master

commit 9fd4e96ebd6424e3ece94b78144972fd8dfeebcb
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Mon Nov 30 19:49:27 2015 -0500

added type checking on integer and boolean
operators

commit 42745db81af5e4deb59d9253595e3c6f81f6178f
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Mon Nov 30 13:52:45 2015 -0500

added float literals and float operators

commit 93bde6b22c8a7e0845200d58b60f4db744809ea6
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Sun Nov 29 23:12:34 2015 -0500

Added .gitignore

commit c605dc9ec9c44d1b6e97b05eda7c7ea28a11ecc1
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Sun Nov 29 22:53:27 2015 -0500

Got assign and print working

commit be26bab9f5075d23a36cd8ce27a08bb04a2b5d6e
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 29 22:46:55 2015 -0500

victory achieved, managed reduce/reduce
errors

commit a14ffc6664502aad2d6520e1fd2c0ea95e740455
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Sun Nov 29 19:38:54 2015 -0500

Removed ignoring white space and fixing
tests

commit 77351c0ed6cdf8bf03c5455b044013e8e87a9d8a
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Sun Nov 29 19:30:05 2015 -0500

Fixed multi statement expected test for
newlines

commit 73904d6d899d7eb2638e220dc76d78d0f82e60f2
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Sun Nov 29 19:29:18 2015 -0500

diff ignores whitespace and fixed newline
issues

commit 3f7f7f8b0a04e3ae4bd5fc288831aa6eaaebacc6
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 29 19:12:24 2015 -0500

fixed statements writing in incorrect order

commit f1dc3be05d5ce81240ce95f284cd7ec7473887ca
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Sun Nov 29 18:52:57 2015 -0500

Working on multi but statements are running
backwards

commit 23edd7a52eab3db3f11a0379558629f17a4ec4d8
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Sun Nov 29 18:41:02 2015 -0500

Now adding missing expected output files for
or and not

commit 54253a4b77d2e1a61a536d1ed5eb3b74f92bbc68
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Sun Nov 29 18:38:19 2015 -0500

Adding source files for or and not

commit 54243bedab7173f570b299ccc95e9d8ee6b099b2
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Sun Nov 29 18:29:24 2015 -0500

Added missing or and not tests that I left
out

commit 0a253eca0c1d6b68f65faa5a3a607b15da2914a5
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Sun Nov 29 17:38:14 2015 -0500

Finished cleaning up merge problems with
Alan's booleans branch

commit b0e34eb3f7b92d690315b6064a6a49c2974a340c
Merge: c9b6f88 33fdb60
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Sun Nov 29 17:13:38 2015 -0500

Merge branch 'booleans' of
github.com:achad4/Accelerator into multi

commit 33fdb608dfe765253a1c1794fc98ebb779ae0b8b
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 29 15:47:51 2015 -0500

    added an and test

commit 4678ebaae2ad46c7eb19e6fe91c196f25f69e560
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 29 15:00:03 2015 -0500

    added a false literal test

commit c9b6f88411171a60b4c6ec045173063098968932
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Wed Nov 25 21:27:00 2015 -0500

    More multiline statement work but still
buggy

commit b14b69e9f06573c71157fa7d029f0a2f79d5febd
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Wed Nov 25 21:09:35 2015 -0500

    Formatting

commit d57e1c5834772ffaa9b53046ecb90717b34b1091
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Wed Nov 25 20:22:33 2015 -0500

    Compiling and taking multiple statements but
; not working

commit da05d25bed729bd80bc39e753ea52f0dc2c8292c
Author: Evan O'Connor <eoconnor@sailthru.com>
Date: Tue Nov 24 21:15:54 2015 -0500

  Beginning work on adding multiple statements

commit 7c7e6c5f02f79e77670449a1499a0357862d989a
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Tue Nov 24 21:10:15 2015 -0500

  added functionality of boolean literals

commit fe390f41306b45244eca3bc67e2beaae7ceaa61e
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Tue Nov 24 19:01:38 2015 -0500

  assign is vacuously working - it's not
throwing errors - will need proper fix once multi-statement
assignments are functional

commit b6efcfe54168670ebc0d4e5ef795a4eeffee2f0a
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Tue Nov 24 18:49:50 2015 -0500

  added source files for assignTest, expoTest,
modTest which I had forgotten to add earlier

commit 2b34e7f8eda89edc7c881f890d081b863e0f0d22
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 22 16:16:36 2015 -0500

  added assign regression test - not currently
working

commit 70e8a906543a568bbea45f185eb6d347606a1d46
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 22 15:52:12 2015 -0500

    added modulus functionality and test to
regression suite

commit 3e5b81d7e8685cea6bbb36a23468a344e38e5720
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 22 15:27:40 2015 -0500

    got exponentiation working and added to the
regression test suite

commit 0092c5ddd5496d472f353974cd705e19591aa971
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 22 14:27:43 2015 -0500

    added division test to regression test
suite, added the sourceFiles folder which I forgot to earlier

commit 9beecbc0cf9ac8377c63570667ae983d7edbf0ba
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 22 14:00:15 2015 -0500

    deleting files in the wrong places

commit fa898448140e50712424a8d0884143d764ded90a
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 22 13:59:20 2015 -0500

finishing touches organizing folders and
updating reg_test.sh

commit e5787c3345fbb13aa74b5170a4cd742796a9f4fc
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 22 13:52:37 2015 -0500

organized files into folders - output
folder, compiledCpp folder, expected output folder, actual output
folder, and updated reg_test.sh to use the folders

commit 4465b2c819435b457e6995aa81fe082b045deabe
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 22 13:09:54 2015 -0500

added multiplication functionality, and
added multiplication test to regression test suite

commit c24033f3dfa067dbe41fad134e81ab8b5e11a126
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 22 12:42:49 2015 -0500

added subtraction test to regression test
suite

commit cf8cf1261355746d23ede3cc8640f6c4edb7f67f
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 22 12:36:15 2015 -0500

got subtraction functioning

commit 64aa58a4612ffae3195e0d3c66fccddea9e354f2
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Nov 20 22:19:24 2015 -0500

almost have assignment working

commit 82e20346deba8663e59ae0843a0976736bac2791
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Fri Nov 20 17:13:01 2015 -0500

no really, addTest working this time

commit 80e01261c77f4c0d66a37a4e2a6ef5362a33b7b6
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Fri Nov 20 16:45:43 2015 -0500

got initial print of arithmetic operators
working

commit 623d621fef06182e8fbb0c66862038c5e035dc31
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Nov 20 15:47:22 2015 -0500

print function working

commit a6354c05e2bfe270cc47ca8f7360ba34fb854c29
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Nov 15 19:31:41 2015 -0500

Minor aesthetic fix

commit eda7a191b2d587eb7db4347a7c65711ee2805a5b
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Nov 15 18:47:23 2015 -0500

Added logging to test suite and renamed to
.cpp extensions

commit c5299c9b70cdeaafb3d669debeafdcf742b7ae9f
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 15 18:02:34 2015 -0500

    ... and I also forgot to add addTest.acc

commit c32a1bf0e0427f648c3b287d72adc0923964c988
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 15 17:59:08 2015 -0500

    forgot to add addTestExpected.c and
addTest.acc

commit dd0e5c1144e5e959715b55d2c599dc72a6088072
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 15 17:02:29 2015 -0500

    added diff test between output of acc
compiler and expected output

commit 2b67abf4ce58af644f648ffbb952233384215e30
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 15 16:53:28 2015 -0500

    added basic regression test script

commit 17c66e614d6eee80081e591d0176b626b95076b7
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Sun Nov 15 16:27:48 2015 -0500

    added c main wrapping function

commit f5f6c68d9be676b957b15a90d3149b7612bf4df9
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sat Nov 14 16:33:53 2015 -0500

  got int + int compilinggit statusgit
statusgit status!

commit 297bdfd6cb348d61ec039ede0581195e7535683d
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sat Nov 14 15:50:43 2015 -0500

  accepting one sast token

commit 62c015d388ed29d6eb6ce46432c2a90d3644f313
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sat Nov 14 14:06:21 2015 -0500

  sast working with binop

commit f3135e5b5a2047f9dafdcbfac4fa4b697c6de57a
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Nov 13 16:00:05 2015 -0500

  print types in sast

commit 4b961663fb207c5030b6f391855435cb95df7f63
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Nov 13 15:53:44 2015 -0500

  sast working with constant int

commit 6ebf68e5b407850c88d7950bf79254e91378fe04
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Nov 13 15:06:08 2015 -0500

multiple statements in AST

commit 30507bc0b843d0485a6528c02b38bdf001599bd5
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Nov 12 20:30:48 2015 -0500

started minimal compiler

commit 33d5c2b17573b174cb1c9b504f6314d48beae9ea
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Nov 10 20:52:34 2015 -0500

working on printing program

commit 78c9914470591df03c29979ba8070c8eb45ebefd
Merge: feb1617 a3c57bb
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Nov 10 18:35:28 2015 -0500

Merge branch 'master' of
https://github.com/achad4/Accelerator

commit a3c57bb27d4af83be5bf1e0be74cffc7ef104367
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Oct 26 15:35:34 2015 -0400

Added NA to scanner

commit 05693edd4496eeb0ca3911797e38aa7bdeb89360
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Oct 26 00:18:39 2015 -0400

Added function calls

commit 3ab9cfcf9683cf1af493b23df53b6121d859b9d9
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Oct 25 23:21:24 2015 -0400

    Tidying up spacing of parser

commit 107e2cdc8a029d98be467ce2e8d7b027779fe40c
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Oct 25 23:17:03 2015 -0400

    Fixed array acces to feature row and column
level access

commit feb16170994a67f4a14a7b13a4b35ff0f594e86e
Merge: 87681b5 7e865d4
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Oct 25 20:37:59 2015 -0400

    Merge branch 'master' of
https://github.com/achad4/Accelerator into avi

commit 87681b5d72eb785083ed090a33519fef5e3125b6
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sun Oct 25 20:37:40 2015 -0400

    tested to see if extraneous directives
compile

commit 7e865d4494e1d30071a3a38ab2ca5f39e182e5b6
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Oct 25 19:07:22 2015 -0400

    Fixed char tokens that were strings

commit d4136520931db7a36bd3be48fb4570ea8df9183f

Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Oct 25 18:33:17 2015 -0400

    Fixed shift/reduce errors by adding MOD
declaration

commit 6d6523e5ed65545287d9e62359a84b1f3409708d
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Oct 25 18:24:06 2015 -0400

    Some refactoring and working on eliminating
shift/reduce. Added matrix access.

commit 192e7357c8a60688aeca84ee7da98375d95e8fe6
Author: evanoconnor <eco2116@columbia.edu>
Date: Sun Oct 25 17:29:48 2015 -0400

    Added more operators and tokens. Many
shift/reduce conflicts at the moment.

commit 7ea0d5f93bc66aa36bc383ddfd521d6fff9291c9
Author: evanoconnor <eco2116@columbia.edu>
Date: Sat Oct 24 21:36:28 2015 -0400

    Changed implementation of characters and
charvectors

commit 886d79114ef58f913fea47dbd24dcc2be4309a2f
Author: evanoconnor <eco2116@columbia.edu>
Date: Sat Oct 24 21:03:16 2015 -0400

    Added matrices

commit 73cc4fe40fc97c751bdc16ba49764aebda8eea87
Author: evanoconnor <eco2116@columbia.edu>
Date: Sat Oct 24 20:36:36 2015 -0400

No more shift/reduce or red/red errors

commit cf4cbc0926de626ca853b867cf0da1c33e3d643b
Author: evanoconnor <eco2116@columbia.edu>
Date: Sat Oct 24 20:22:33 2015 -0400

Added return. Working on eliminating
shift/reduce errors possibly due to break/next

commit 21c8610229e005e854ef8489d2229cb47d4b311c
Author: evanoconnor <eco2116@columbia.edu>
Date: Sat Oct 24 19:23:03 2015 -0400

Fixed bug in for loops

commit 52d01d21a3b7b95f1376b700232099f413f190bc
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sat Oct 24 17:03:16 2015 -0400

working on loop stmts

commit 442689cadf65464de79155827e8bce2fb7d84930
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sat Oct 24 15:59:33 2015 -0400

noelse

commit b62dcee2ce2efa305bfdafcd5789276a40e49dbb
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sat Oct 24 15:40:55 2015 -0400

basic parsing and grammar working

commit d04c1056f94c945baae45ac8552285ef2112de43
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sat Oct 24 14:06:21 2015 -0400

    removied conflicts file

commit f8f306f1b2a38d269092e6d6c41dc71ad2716481
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sat Oct 24 14:01:21 2015 -0400

    updated directory structure

commit 8a24c44f69ae172b5ac4540778ed826f07b36a8e
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sat Oct 24 10:48:17 2015 -0400

    working on literal logic

commit 12a875da3501afe724e67de183b86c24172da5b6
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sat Oct 24 09:39:32 2015 -0400

    program now conists of a list of stmnts and
a list of functions

commit 5e4278d53693dfba0d3510630bb35dec66fc3089
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Sat Oct 24 09:27:15 2015 -0400

    got rid of shift/reduce conflicts by not
allowing vdecs (don't exist in r anyway)

commit 73aa512275f9cf5632d88fe7f4f4972fa6d95fee
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Fri Oct 23 21:37:20 2015 -0400

alan's work so far

commit 2127a658733f5bceaeada8f4a9edc107f6428eb9
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Oct 23 16:51:07 2015 -0400

print statements for testing

commit bff0d1688467f0af58e4841c31bbd150b08db0a7
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Oct 23 16:34:57 2015 -0400

got grammar compiling

commit b81c7c26c67ff61fd211ab1558fb44b3a8c3b465
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Oct 23 15:20:24 2015 -0400

more grammar

commit 1cda33e1dbfa551fa19fea1e1197ade1d8be439b
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Fri Oct 23 15:19:32 2015 -0400

more grammar

commit d77060f6bec4459a26038aeb58a7019fe3eac0a0
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Oct 22 21:21:02 2015 -0400

gramma

commit dc58bc25801eb430bb3088f97b5d8c59d0999cbc

Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Oct 22 21:18:58 2015 -0400

    grammar shit

commit 489d38fd159099d56a2e7305ad0420c8ddfbff9a
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Oct 22 19:29:22 2015 -0400

    makefile

commit 61729ae9d9d375945122ad65319e5ccb0db427fc
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Oct 22 19:21:02 2015 -0400

    ast and parser

commit 58afdf41cd8973be6b7ecd8d7fe692c6c1e66be8
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Oct 22 19:19:47 2015 -0400

    copied in ast and parser

commit c0632caae115b42a3d53e5d65bb1b5b49ed4f1d8
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Oct 22 19:00:20 2015 -0400

    parser work

commit d86609006c39aed6a0736474e49b2f4ee4c32683
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Thu Oct 22 18:06:11 2015 -0400

    forgot to add scanner.mll

commit de65c1038bf1c3897dc3ac52d29f1ff674050bda
Author: Alan Phillip McNaney
<apm2144@columbia.edu>
Date: Thu Oct 22 17:38:46 2015 -0400

    Alan testing git commits

commit 7cc9a9fcecbfea209d88022f0eee065f929f6463
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Oct 19 18:52:27 2015 -0400

    Removed old copies of .tex and .pdf
proposals

commit e28ca4e0871992d2fa5de405660ea879adb8d2b5
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Oct 19 18:51:25 2015 -0400

    Removed old proposal pdf and reorganized
directories

commit 0835e0ea64156297ed6305dd4127d7e86a83df9e
Author: evanoconnor <eco2116@columbia.edu>
Date: Mon Oct 19 18:48:47 2015 -0400

    Added completed proposal

commit 908d50fcd4937ff6f090aa0819029965b69b05a9
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Oct 15 20:07:41 2015 -0400

    fixed bug

commit cb3b3688ab0113f5658387cd7a85613892357631

Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Oct 15 19:53:24 2015 -0400

    random numbers in input matrices

commit 5408af76a2467478563a208b294768d9bd803a14
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Oct 15 08:40:09 2015 -0400

    rename

commit d43003c12075a36aad43653d3bff5b453d9afeee
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Thu Oct 15 08:38:18 2015 -0400

    project proposal

commit 968c8d01fd72012d61949fe5e271a2ff529ac904
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Oct 13 20:36:26 2015 -0400

    reference for matrix mult

commit 241f98e1341ac4b347400a12f7a3c8446d249f8d
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Oct 13 20:30:50 2015 -0400

    moving files

commit a8a17a2d4af3cea1e3478da6f856e46fb52dbd8e
Author: Avi Chad-Friedman <ajc2212@columbia.edu>
Date: Tue Oct 13 20:29:01 2015 -0400

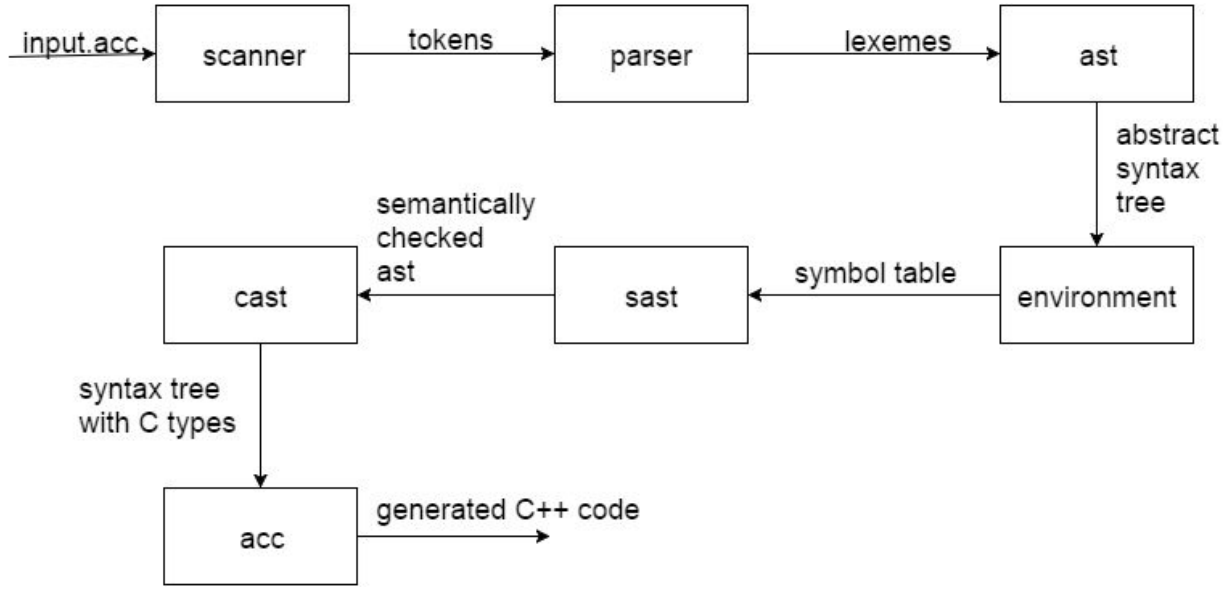
    example matrix mult in openmp

commit 8149037d5208a82a4418637b00e791acd2cd5f10
Author: Avi Chad-Friedman
<achad4@users.noreply.github.com>
Date: Tue Oct 13 20:26:35 2015 -0400

Initial commit

5.) Architectural Design

5.1) Block Diagram of Translator



5.2) Interfaces Between Components

- Scanner to Parser: The scanner takes in a character stream and creates a stream of tokens, which it passes to the parser.
- Parser to AST: The parser intakes a string of tokens which it forms into semantically correct expressions and statements. Those are passed to the Abstract Syntax Tree
- Ast to Environment: The AST takes in a stream of expressions and statements, and build them together into an abstract syntax tree.
- Environment to Sast: The environment receives an abstract syntax tree, which it traverses collecting variable names and the types they contain into a symbol table.

- Sast to Cast: The Sast intakes an abstract syntax tree with symbol tables attached, and does a semantic check on all nodes in the syntax tree, ensuring type correctness.
- Cast to Acc: The cast intakes a semantically and syntactically checked abstract syntax tree, and uses it to generate OpenMP enabled C++, our target language.

5.3) Who Implemented Components

Given our slightly understaffed team, it was again a group effort on most components.

- Scanner - All
- Parser - All
- Ast - all
- Environment - Avi and Evan
- Sast - All
- Cast - All
- Acc - All

6.)Test Plan

6.1) Representative Source Programs and Target Language Counterparts

6.1.1) Representative Program 1: Matrix Multiplication

6.1.1.a) Source Language Code:

Code of matrixMultTest.acc

```
1 f <- function(){
2 a <- matrix(c(1,2,3,4), nrow=2,ncol=2)
3 b <- matrix(c(4,3,2,1), nrow=2,ncol=2)
4 a * b
5 }
6 print(f())
```

6.1.1.b) Target Language Code

Code of matrixMultTest.cpp

```
1 #include<iostream>
2 #include<stdio.h>
```

```
 3 #include<math.h>
 4 #include<vector>
 5 #include<string>
 6 #include<string.h>
 7 using namespace std;
 8
 9      vector<vector<int> > matrix_add(vector<vector<int> > a, vector<vector<int > >
b){
10      vector<vector<int> > c;
11      for(int i = 0; i < a.size(); i++){
12              vector<int> row;
13              for(int j = 0; j < a[0].size(); j++){
14              row.push_back(a[i][j] + b[i][j]);
15              }
16              c.push_back(row);
17      }
18      return c;
19      }
20
21      vector<vector<int> > matrix_mult(vector<vector<int> > a, vector<vector<in t> >
b){
22      vector<vector<int> > c;
23      int i,j,k;
24      #pragma omp parallel shared(a,b,c) private(i,j,k)
25      {
26      #pragma omp for  schedule(static)
27      for (i=0; i<a.size(); i=i+1){
28              vector<int> row;
29              for (j=0; j<b[0].size(); j=j+1){
30                      row.push_back(0);
31                      for (k=0; k<b.size(); k=k+1){
32                      row[j] += ((a[i][k])*(b[k][j]));
33                      }
34              }
35              c.push_back(row);
36              }
37      }
38      return c;
39      }
40      void print_matrix(vector<vector<int> > a){
41      for(int i = 0; i<a.size(); i++){
42              cout<<endl;
43              for(int j = 0; j<a[0].size(); j++){
44              cout << a[i][j] << " ";
```

```
45              }
46     }
47     }
48 vector<vector<int> > f(){
49 int aHolder[] = {1, 2, 3, 4};
50 vector<int> aVector (aHolder, aHolder + 2 / sizeof(int));
51 vector<vector<int> > a (2);
52 int acount=0;
53 for(int i=0; i<2; i++) {
54 a[i].resize(2);
55 for(int j=0; j<2; j++) {
56 a[i][j] = aHolder[acount++];
57 }
58 };
59 int bHolder[] = {4, 3, 2, 1};
60 vector<int> bVector (bHolder, bHolder + 2 / sizeof(int));
61 vector<vector<int> > b (2);
62 int bcount=0;
63 for(int i=0; i<2; i++) {
64 b[i].resize(2);
65 for(int j=0; j<2; j++) {
66 b[i][j] = bHolder[bcount++];
67 }
68 };
69 return (matrix_mult(a, b));
70 }
71 int main(){
72 print_matrix(f());
73 }
```

6.1.1.c) Why this test case?

This test case is chosen to display one of the primary focuses of the Accelerator language, to allow access to parallel computing resources in an R context. It is also a good example of the level of complexity which the programmer in Accelerator, familiar with R syntax, is being shielded from.

6.1.2) Representative Program 2: Testing Functions

6.1.2.a) Source Language Code

Code of funcTest.acc

```
1 foo <- function(i=0, j=0.0){
2       a <- i
```

```
 3       print(a)
 4       return(3)
 5 }
 6 bar <- function(i=0) {
 7       i<-3
 8       i<-5
 9       return(i)
10 }
11 baz <- function() {
12       print("hello")
13       return("world")
14 }
15 a <- foo(4, 1.1)
16 print(a)
17 b <- bar(100)
18 print(b)
19 c <- baz()
20 print(c)
```

6.1.2.b) Target Language Code

Code of funcTest.cpp

```
 1 #include<iostream>
 2 #include<stdio.h>
 3 #include<math.h>
 4 #include<vector>
 5 #include<string>
 6 #include<string.h>
 7 using namespace std;
 8
 9      vector<vector<int> > matrix_add(vector<vector<int> > a, vector<vector<int > >
b){
10      vector<vector<int> > c;
11      for(int i = 0; i < a.size(); i++){
12              vector<int> row;
13              for(int j = 0; j < a[0].size(); j++){
14              row.push_back(a[i][j] + b[i][j]);
15              }
16              c.push_back(row);
17      }
18      return c;
19      }
20
```

```
 21     vector<vector<int> > matrix_mult(vector<vector<int> > a, vector<vector<in t> >
b){
 22     vector<vector<int> > c;
 23     int i,j,k;
 24     #pragma omp parallel shared(a,b,c) private(i,j,k)
 25     {
 26     #pragma omp for  schedule(static)
 27     for (i=0; i<a.size(); i=i+1){
 28             vector<int> row;
 29             for (j=0; j<b[0].size(); j=j+1){
 30                     row.push_back(0);
 31                     for (k=0; k<b.size(); k=k+1){
 32                     row[j] += ((a[i][k])*(b[k][j]));
 33                     }
 34             }
 35             c.push_back(row);
 36             }
 37     }
 38     return c;
 39     }
 40     void print_matrix(vector<vector<int> > a){
 41     for(int i = 0; i<a.size(); i++){
 42             cout<<endl;
 43             for(int j = 0; j<a[0].size(); j++){
 44             cout << a[i][j] << " ";
 45             }
 46     }
 47     }
 48 int foo(int i=0,float j=0.){
 49 int a = i;
 50 cout << a;
 51 cout << endl;
 52 return (3);
 53 }
 54 int bar(int i=0){
 55 i = 3;
 56 i = 5;
 57 return (i);
 58 }
 59 string baz(){
 60 cout <<  "hello" ;
 61 cout << endl;
 62 return ( "world" );
 63 }
```

```
64 int main(){
65 int a = foo(4, 1.1);
66 cout << a;
67 cout << endl;
68 int b = bar(100);
69 cout << b;
70 cout << endl;
71 string c = baz();
72 cout << c;
73 cout << endl;
74 }
```

6.1.2.a) Why this test case?

This test case is representative of how functions are handled in Accelerator.

6.1.3) Representative Program 3: Greatest Common Divisor Function

6.1.3.a) Source Language Code

Code of gcdTest.acc

```
1 gcd <- function(a=0,b=0, t=0) {
2 while(b != 0) {
3 t <- b
4 b <- a %% b
5 a <- t
6 }
7 return(a)
8 }
9 d <- gcd(15,10,0)
10 print(d)
```

6.1.3.b) Target Language Code

Code of gcdTest.cpp

```
1 #include<iostream>
2 #include<stdio.h>
3 #include<math.h>
4 #include<vector>
5 #include<string>
6 #include<string.h>
7 using namespace std;
8
```

```
 9      vector<vector<int> > matrix_add(vector<vector<int> > a, vector<vector<int > >
b){
 10     vector<vector<int> > c;
 11     for(int i = 0; i < a.size(); i++){
 12             vector<int> row;
 13             for(int j = 0; j < a[0].size(); j++){
 14             row.push_back(a[i][j] + b[i][j]);
 15             }
 16             c.push_back(row);
 17     }
 18     return c;
 19     }
 20
 21     vector<vector<int> > matrix_mult(vector<vector<int> > a, vector<vector<in t> >
b){
 22     vector<vector<int> > c;
 23     int i,j,k;
 24     #pragma omp parallel shared(a,b,c) private(i,j,k)
 25     {
 26     #pragma omp for  schedule(static)
 27     for (i=0; i<a.size(); i=i+1){
 28             vector<int> row;
 29             for (j=0; j<b[0].size(); j=j+1){
 30                     row.push_back(0);
 31                     for (k=0; k<b.size(); k=k+1){
 32                     row[j] += ((a[i][k])*(b[k][j]));
 33                     }
 34             }
 35             c.push_back(row);
 36             }
 37     }
 38     return c;
 39     }
 40     void print_matrix(vector<vector<int> > a){
 41     for(int i = 0; i<a.size(); i++){
 42             cout<<endl;
 43             for(int j = 0; j<a[0].size(); j++){
 44             cout << a[i][j] << " ";
 45             }
 46     }
 47     }
 48 int gcd(int a=0,int b=0,int t=0){
 49 while (b != 0){
 50 t = b;
```

```
51 b = a % b;
52 a = t;
53 }
54 return (a);
55 }
56 int main(){
57 int d = gcd(15, 10, 0);
58 cout << d;
59 cout << endl;
60 }
```

6.1.3.a) Why this test case?

This test shows the Turing complete nature of Accelerator with a classic function.

6.2) Test Suite

We worked in a regression test driven development style. With each new feature we wished to add to the language, we would add a test to our regression test suite, which would of course fail, and then attempt to get it to pass. Once the tests passed, we would move on to adding new features.

6.2.1) Regression Test Shell Script

Below is the code of our regression test suite.

Code of reg_test.sh

```
 1 #!/bin/bash
 2 make clean && make
 3
 4 # Count number of tests and failures
 5 declare -i count
 6 declare -i failures
 7 count=0
 8 failures=0
 9
10 runTest() {
11     ./acc < sourceFiles/$1.acc > compiledCpp/$1.cpp
12     g++ -o executables/$1 compiledCpp/$1.cpp
13     ./executables/$1 > output/$1.txt
14     count+=1
15     if diff "output/"$1".txt" "expected/"$1"Expected.txt" > /dev/null; then
16     echo $1 passed
17     else
```

```
18      echo $1 failed
19      failures+=1
20      fi
21 }
22
23 runTestOpenMP() {
24      ./acc < sourceFiles/$1.acc > compiledCpp/$1.cpp
25      clang-omp++ -o executables/$1 compiledCpp/$1.cpp -fopenmp
26      ./executables/$1 > output/$1.txt
27      count+=1
28      if diff "output/"$1".txt" "expected/"$1"Expected.txt" > /dev/null; then
29      echo $1 passed
30      else
31      echo $1 failed
32      failures+=1
33      fi
34 }
35
36 if [ $# -eq 1 ]
37   then
38      runTest $1
39      exit
40 fi
41
42 if [ $# -eq 2 ]
43      then
44      if [ $2 == "--openmp" ]
45      then
46              runTestOpenMP $1
47              exit
48      fi
49 fi
50
51 runTest "helloWorldTest"
52 runTest "addTest"
53 runTest "subTest"
54 runTest "multTest"
55 runTest "divTest"
56 runTest "expoTest"
57 runTest "modTest"
58 runTest "assignTest"
59 runTest "trueLitTest"
60 runTest "falseLitTest"
61 runTest "andTest"
```

```
62 runTest "orTest"
63 runTest "notTest"
64 runTest "comparisonTest"
65 runTest "multiStatementTest"
66 runTest "floatLitTest"
67 runTest "floatOpTest"
68 runTest "assignAndPrintTest"
69 runTest "vectorTest"
70 runTest "ifElseTest"
71 runTest "forTest"
72 runTest "matrixTest"
73 runTest "stringTest"
74 runTest "funcTest"
75 runTest "eqNeqTest"
76 runTest "whileTest"
77 runTest "gcdTest"
78 runTest "scopeTest"
79 runTest "matrixAdd"
80 runTest "matrixMultTest"
81
82 echo =====================================================
83 echo Results
84 echo =====================================================
85 echo Test Suite finished
86 echo Failed Tests: $failures
87 echo Total Tests: $count
88 make clean
```

6.2.2) Regression Test Shell Script Output

Command Line Output of Running ./reg_test.sh

```
$ ./reg_test.sh
rm -f acc parser.ml parser.mli scanner.ml testall.log \
    *.cmo *.cmi *.out *.diff
ocamlc -c ast.ml
ocamlyacc parser.mly
ocamlc -c parser.mli
ocamlc -c parser.ml
ocamllex scanner.mll
```

```
90 states, 4376 transitions, table size 18044 bytes
ocamlc -c scanner.ml
ocamlc -c environment.ml
ocamlc -c sast.ml
ocamlc -c cast.ml
ocamlc -c acc.ml
ocamlc -o acc ast.cmo parser.cmo scanner.cmo environment.cmo
sast.cmo cast.cmo acc.cmo
helloWorldTest passed
addTest passed
subTest passed
multTest passed
divTest passed
expoTest passed
modTest passed
assignTest passed
trueLitTest passed
falseLitTest passed
andTest passed
orTest passed
notTest passed
multiStatementTest passed
floatLitTest passed
floatOpTest passed
assignAndPrintTest passed
vectorTest passed
ifElseTest passed
forTest passed
matrixTest passed
stringTest passed
funcTest passed
eqNeqTest passed
whileTest passed
gcdTest passed
scopeTest passed
matrixAdd passed
matrixMultTest passed
====================================================
```

```
Results
==================================================
Test Suite finished
Failed Tests: 0
Total Tests: 29
rm -f acc parser.ml parser.mli scanner.ml testall.log \
    *.cmo *.cmi *.out *.diff
```

6.3) What Kind of Automation

The testing was drivent by a shell script. It compiles the ./acc compiler, then carries out the following set of actions on each .acc test in the sourceCode directory included in the tar.gz

1.) Ingest the .acc test file into the ./acc compiler, and output the result to the /compiledCpp folder
2.) Compile the ./cpp generated in step 1 with g++, creating an executable for the test in the /executables folder
3.) Run the executable, and capture it's output into a .txt file in the /output folder
4.) Compare that generated output, using diff, with an expected output .txt file in the /expected folder
5.) If the output and the expected txt files are identical, print out "<testname> passed" to standard out
6.) If the output and the expected txt files differ, print out "<testname> failed" to standard out

After running all tests, the shell script displays a total of the failing tests, and a count of all tests run.

6.4) Who Did What?

As in all sections, it was a group effort. The original test script was developed by Alan McNaney, and was edited and greatly improved by Evan O'Connor. All members wrote and integrated tests into the regression test suite.

7.)Lessons Learned

7.1) Avi Jacob Chad-Friedman

7.1.1) Lessons Learned

Cross the river; build the bridge. Professor Edwards told us this this during our first demo, but the weight of the lesson did not dawn on me until the few days before the final presentation. We avoided handling scoping and environments early on in favor of building up other functionality, like all of the various operators and data types. This made integrating an environment a behemoth task, one that nearly took down our whole project. It was nearly impossible to debug scoping and symbol table issues with so much functionality already built around it.

7.1.2) Advice

Never fall in love with your own ideas. Human beings have a hard time letting go of sunk costs. Know this and know that a convenient hack will always come back to haunt you. If a conceptual block seems hard to overcome at the moment, it will only become more daunting when it precludes you from implementing the next stage of your compiler.

7.2) Evan Charles O'Connor

7.2.1) Lessons Learned

We should have put in more time earlier. It would have been great to get ahead early. There are always going to be hiccups and roadblocks, and it would have been great to have time to come at those challenges with time to spare. This is a project that can easily get away from you, even if you think you're putting in a lot of time. Getting very little sleep for the last week isn't a position you want to be in, particularly with other projects and finals piling on top of this project.

7.2.2) Advice

Speak your mind. Ideas that go unchallenged are prone to bugs. Even if you think that a teammate better understands a certain section of code, or, even, is a better programmer in general, try your hardest to find flaws in his or her implementation. Don't accept their approach just because it sounds nice; ask them to walk you through their thinking. You'll discover something important in this verbal dissection surprisingly often.

7.3) Alan Phillip McNaney

7.3.1) Lessons Learned

Working well with your team is huge. This is a large project, and having people who are hardworking and easy to get along with is a must. Keeping to a regular schedule is important. Other work is going to get in the way, so having a structural, schedule based method of bringing yourself back to task with regularity ensured that we

got as far as we did. Mainly, I learned a great respect for those who have come before, putting countless hours in to provide the tools we take for granted.

7.3.2) Advice

I know that everyone says it, but start early. Get printing working as soon as you can, otherwise how do you know anything's working? Be careful when selecting your language - it is *very* easy to bite off… not necessarily more than you can chew, but more than you can comfortably chew.

8.)Appendix

8.1) Translator Files

scanner.mll

```
{

        open Parser
 open Lexing
}



 let dig = ['-']?['0'-'9']
 let frac = '.' dig*
 let whitespace = [' ' '\t']
 let flt = dig* frac
 rule token = parse
        | whitespace    { token lexbuf }

   | "NA"          { NA }
   | "true" as lit {
TRUE(bool_of_string lit) }
   | "false" as lit{
FALSE(bool_of_string lit) }
   | "if"          { IF }
   | "else"        { ELSE }
   | "for"         { FOR }
   | "while"       { WHILE }
```

```
  | ":"          { RANGE }
  | "function"    { FUNCTION }
  | "return"      { RETURN }
  | "c("          { VECTSTART }
  | "matrix("     { MATRIXSTART }
  | "nrow"        { NROW }
  | "ncol"        { NCOL }
  | "="           { EQSING }
  | "in"          { IN }
        | ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9'
'_']* as lxm { ID(lxm) }
  | '\"' [^'\"']* '\"' { let
str = lexeme lexbuf in
                      STRING (String.sub
str 1 (String.length str - 2)) }
        | ','          { COMMA }
        | '('          { LPAREN }
          | ')'          { RPAREN }
  | '{'          { LBRACE }
  | '}'          { RBRACE }
        | dig+ as lit   { INT(int_of_string lit) }
  | flt as lit    { FLOAT(float_of_string lit)
}
  | '\n'         { DLIN }
  | '+'          { PLUS }
  | '-'          { MINUS }
  | '*'          { MULT }
  | '/'          { DIV }
  | '^'          { EXPO }
  | "<-"         { ASSIGN }
  | "%%"         { MOD }
  | "&&"         { AND }
  | "||"         { OR }
  | '!'          { NOT }
  | "=="         { EQ }
  | "!="         { NEQ }
  | '['          { LBRAC }
  | ']'          { RBRAC }
```

```
  | eof           { EOF }
```

parser.mly

```
%{ open Ast %}



%token EOF, DLIN, PLUS, MINUS, MULT, DIV, EXPO,
MOD, RBRACE, LBRACE, NA
%token LPAREN, RPAREN, COMMA, ASSIGN, AND, OR,
NOT, IF, ELSE, VECTSTART
%token MATRIXSTART, NROW, NCOL, EQSING, EQ, FOR,
IN, RANGE, LBRAC, RBRAC
%token FUNCTION, RETURN, NEQ, WHILE
%token <int> INT
%token <float> FLOAT
%token <string> ID
%token <string> STRING
%token <bool> TRUE FALSE



%left PLUS MINUS

%left MULT DIV
%left MOD
%left AND OR
%right ASSIGN NOT EXPO



%start program
```

```
%type <Ast.program> program

%%

program :
 | EOF
      { ([],[]) }
 | func_def_list
      { ($1, []) }
 | stmt_list
      { ([], $1) }
 | func_def_list stmt_list EOF
      { ($1, $2) }

func_stmt_list:
 | func_stmt
      { [$1] }
 | func_stmt_list func_stmt
      { $2 :: $1 }

block:
 | LBRACE DLIN stmt_list RBRACE DLIN
      { Block(List.rev $3) }

func_block:
 | LBRACE DLIN func_stmt_list RBRACE DLIN
      { Block(List.rev $3) }
```

```
stmt_list:
 | stmt
      { [$1] }
 | stmt_list stmt
      { $2 :: $1 }


stmt:
 | expr DLIN
      { Expr($1) }
 | IF LPAREN bool_expr RPAREN block ELSE block
      { If($3, $5, $7) }
 | FOR LPAREN ID IN expr RANGE expr RPAREN
block      { For($3, $5, $7, $9) }
 | WHILE LPAREN compare_expr RPAREN block
      { While ($3, $5) }


func_def_list:
 | func_def
      { [$1] }
 | func_def_list func_def
      { $2 :: $1 }


func_def:
  | ID ASSIGN FUNCTION LPAREN formals_opt
RPAREN func_block { FunctionDef($1, $5, $7) }


func_stmt:
 | stmt
          { $1 }
```

```
 | RETURN LPAREN expr RPAREN DLIN
          { Return($3) }



expr:
 | ID
          { Id($1) }
 | compare_expr
          { $1 }
 | expr MULT expr
          { Mult($1, $3) }
 | expr PLUS expr
          { Add($1, $3) }
 | expr MINUS expr
          { Sub($1, $3) }
 | expr DIV expr
          { Div($1, $3) }
 | expr EXPO expr
          { Expo($1, $3) }
 | expr MOD expr
          { Mod($1, $3) }
 | literal
          { $1 }
 | bool_expr
          { $1 }
 | string_expr
          { $1 }
 | ID ASSIGN VECTSTART vect_opt RPAREN
          { Vector($1, $4) }
 | ID ASSIGN MATRIXSTART VECTSTART vect_opt
RPAREN COMMA

   NROW EQSING expr COMMA NCOL EQSING expr
RPAREN          { Matrix($1, $5, $10, $14) }
 | ID LPAREN actuals_opt RPAREN
          { FuncCall($1, $3) }
 | ID ASSIGN expr
          { Assign($1, $3) }
```

```
 | ID LBRAC expr RBRAC
        { VectAcc($1, $3) }
 | ID LBRAC expr RBRAC LBRAC expr RBRAC
        { MatrixAcc($1, $3, $6) }



compare_expr:
 | eq_expr EQ eq_expr
        { Eq($1, $3) }
 | eq_expr NEQ eq_expr
        { Neq($1, $3) }



eq_expr:
 | literal
        { $1 }
 | bool_expr
        { $1 }
 | string_expr
        { $1 }
 | ID
        { Id($1) }



vect_opt:
 | /* nothing */
        { [] }
 | lits
        { List.rev $1 }
 | bool_lits
        { List.rev $1 }



literal:
```

```
 | INT
        { IntLit($1) }
 | FLOAT
        { FloatLit($1) }




bool_literal:
 | TRUE
        { BoolLit(true) }

 | FALSE
        { BoolLit(false) }




lits:

 | literal
        { [$1] }
 | lits COMMA literal
        { $3 :: $1 }
 | lits COMMA NA
        { Na :: $1 }



bool_lits:
 | bool_expr
        { [$1] }
 | bool_lits COMMA bool_expr
        { $3 :: $1 }
 | bool_lits COMMA NA
        { Na :: $1 }
```

```
formals_opt:
 | /* nothing */
        { [] }
 | formals_list
        { List.rev $1 }



formals_list:
 | formal_def
        { [$1] }
 | formals_list COMMA formal_def
        { $3 :: $1 }



formal_def:
 | ID EQSING expr
        { FormalDef($1, $3) }


actuals_opt:
 | /* nothing */
        { [] }
 | actuals_list
        { List.rev $1 }



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

```
bool_expr:
 | bool_literal
          { $1 }
 | bool_expr AND bool_expr
          { And($1, $3) }
 | bool_expr OR bool_expr
          { Or($1, $3) }
 | NOT bool_expr
          { Not($2) }



string_expr:
 | string_data
          { $1 }



string_data:
 | STRING
          { StringLit($1) }
```

ast.ml

```
type op =

 | Eq
 | Neq
 | Add
 | Sub
 | Mult
 | Div
 | Expo
 | Mod
```

```
 | Assign
 | And
 | Or
 | Not



type expr =
 | Na
 | Id of string
        | IntLit of int
 | BoolLit of bool
 | FloatLit of float
 | StringLit of string
 | Vector of string * expr list
 | VectAcc of string * expr
 | Matrix of string * expr list * expr * expr
 | MatrixAcc of string * expr * expr
 | Eq of expr * expr
 | Neq of expr * expr
 | Add of expr * expr
 | Sub of expr * expr
 | Mult of expr * expr
 | Div of expr * expr
 | Expo of expr * expr
 | Mod of expr * expr
        | FuncCall of string * expr list

        | Assign of string * expr
 | And of expr * expr
 | Or of expr * expr
 | Not of expr
 | FormalDef of string * expr



type stmt =

        | Expr of expr
```

```
 | Block of stmt list
 | If of expr * stmt * stmt
 | For of string * expr * expr * stmt
 | While of expr * stmt
 | Return of expr


type func_def =

 | FunctionDef of string * expr list * stmt


type program = func_def list * stmt list
```

environment.ml

```
module VarMap = Map.Make(String);;
module FuncMap = Map.Make(String);;


exception NoEnvironmentException;;
exception UnassignedVarException of string;;
exception IncorrectTypeException;;


type op =

 | Eq
 | Neq
 | Add
```

```
 | Sub
 | Mult
 | Div
 | Expo
 | Mod
 | Assign
 | And
 | Or
 | Not



type t =

 | String
 | Int
 | Float
 | Bool
 | Vector
 | Matrix
 | Na



type expr =
 | Na

 | Id of string
 | IntLit of int
 | BoolLit of bool
 | FloatLit of float
 | StringLit of string
 | Vector of string * expr list
 | VectAcc of string * expr
 | Matrix of string * expr list * expr * expr
 | MatrixAcc of string * expr * expr
 | Eq of expr * expr
 | Neq of expr * expr
 | Add of expr * expr
```

```
 | Sub of expr * expr
 | Mult of expr * expr
 | Div of expr * expr
 | Expo of expr * expr
 | Mod of expr * expr
 | FuncCall of string * expr list

 | Assign of string * expr
 | Update of string * expr
 | And of expr * expr
 | Or of expr * expr
 | Not of expr
 | FormalDef of string * expr * t



type stmt =

 | Expr of expr
 | Block of stmt list
 | If of expr * stmt * stmt
 | For of string * expr * expr * stmt
 | While of expr * stmt
 | Return of expr



type func_def =

 | FunctionDef of string * expr list * stmt



type program = func_def list * stmt list



type environment = {
```

```
  symb_tbl_stk: t VarMap.t list;

  func_tbl: t FuncMap.t;
  func_tbl_formals: t list FuncMap.t;
}



let init_env = {
  symb_tbl_stk = VarMap.empty::[];
  func_tbl = FuncMap.empty;
  func_tbl_formals = FuncMap.empty;
}



let find_type id env =
 let rec search_scope_lvl lvl =
 match lvl with
   | [] -> raise (UnassignedVarException id)
   | top :: rest ->
     if VarMap.mem id top then
       VarMap.find id top
     else
       search_scope_lvl rest
 in search_scope_lvl env.symb_tbl_stk



let rec type_match env = function
 | StringLit(s) -> String
 | IntLit(i) -> Int
 | FloatLit(f) -> Float
 | BoolLit(b) -> Bool
 | Na -> Na
 | Matrix(s,el,e1,e2) -> Matrix
 | Vector(s,el) -> Vector
 | Id(id) -> find_type id env
```

```
 | FormalDef(id, e, t) -> type_match env e
 | FuncCall(s, el) ->
     if (s = "print") then
       Na
     else if FuncMap.mem s env.func_tbl then
       FuncMap.find s env.func_tbl
     else
       failwith "Function does not exist"
 | _ -> Matrix



let find_type_top id env =
 let rec search_scope_lvl lvl =
 match lvl with
   | [] -> raise (UnassignedVarException id)
   | top :: rest ->
     if VarMap.mem id top then
       VarMap.find id top
     else
       type_match env Na
 in search_scope_lvl env.symb_tbl_stk



let rec type_of_stmt env = function
 | Expr(e) -> type_match env e
 | Block(sl) -> let last_stmt = List.nth sl
(List.length sl - 1) in
                  type_of_stmt env last_stmt
 | If(e,s1,s2) -> Na
 | For(s1,e1,e2,s2) -> Na
 | While(e,s) -> Na
 | Return(e) -> type_match env e



let reassign_symb_tbl_stk stk func forms = {
```

```
   symb_tbl_stk = stk;
   func_tbl = func;
   func_tbl_formals = forms;
}



let push_env_scope env =

   reassign_symb_tbl_stk
(VarMap.empty::env.symb_tbl_stk) env.func_tbl env.func_tbl_formals



let pop_env_scope env =

  match env.symb_tbl_stk with

   | [] -> raise NoEnvironmentException

   | curr::rest ->

       reassign_symb_tbl_stk rest env.func_tbl
env.func_tbl_formals



let assign_current_scope fname var vtype env =
 let curr, rest =
   ( match env.symb_tbl_stk with
     | curr :: rest -> curr, rest
     | [] -> raise NoEnvironmentException )
in
   let updated = VarMap.add var vtype curr in
 reassign_symb_tbl_stk (updated::rest)
env.func_tbl env.func_tbl_formals
```

```
let assign_current_scope var vtype env =
 let curr, rest =
   ( match env.symb_tbl_stk with
     | curr :: rest -> curr, rest
     | [] -> raise NoEnvironmentException )
in
   let updated = VarMap.add var vtype curr in
 reassign_symb_tbl_stk (updated::rest)
env.func_tbl env.func_tbl_formals



let rec scope_expr_detail env = function
 | Ast.Na -> Na, env
 | Ast.Id(s) -> Id(s), env
 | Ast.IntLit(i) -> IntLit(i), env
 | Ast.BoolLit(b) -> BoolLit(b), env
 | Ast.FloatLit(f) -> FloatLit(f), env
 | Ast.StringLit(s) -> StringLit(s) , env
 | Ast.Matrix(s, el, e1, e2) ->
     let head = List.hd el in
     let mtype = type_match env (fst
(scope_expr_detail env head)) in
     let new_env = assign_current_scope s
Matrix env in
     let s_type = s ^ "type" in
     let new_env = assign_current_scope s_type
mtype new_env in
     let helper e = fst (scope_expr_detail env
e) in
     Matrix(s, List.map helper el,
fst(scope_expr_detail env e1), fst(scope_expr_detail env e2)),
new_env
 | Ast.Assign(s,e) ->
   let e1 = scope_expr_detail env e in
   let t = type_match env (fst e1) in
   let old_t = find_type_top s env in
```

```
   if (old_t == Na) then (
     let new_env = assign_current_scope s t env
in
      Assign(s,fst(e1)), new_env
   )
   else if (old_t = t) then (
     Update(s, fst(e1)), env
   )
   else
   failwith "Cannot cast types"
 | Ast.Vector(s, el) ->
     let head = List.hd el in
     let e1 = scope_expr_detail env head in
     let t = type_match env (fst e1) in
     let new_env = assign_current_scope s t env
in
     let helper e = fst (scope_expr_detail env
e) in
     Vector(s, List.map helper el), new_env
 | Ast.VectAcc(s, expr) -> VectAcc(s,
fst(scope_expr_detail env expr)), env
 | Ast.MatrixAcc(s, e1, e2) ->

     MatrixAcc(s, fst(scope_expr_detail env
e1), fst(scope_expr_detail env e2)), env
 | Ast.Eq(expr1,expr2) ->
     let e1, v1 = scope_expr_detail env expr1
in
     let e2, v2 = scope_expr_detail v1 expr2 in
     Eq(e1, e2), v2
 | Ast.Neq(expr1,expr2) ->
     let e1, v1 = scope_expr_detail env expr1
in
     let e2, v2 = scope_expr_detail v1 expr2 in
     Neq(e1, e2), v2
 | Ast.Add(expr1,expr2) ->
```

```
    let e1, v1 = scope_expr_detail env expr1
in
    let e2, v2 = scope_expr_detail v1 expr2 in
      Add(e1, e2), v2
 | Ast.Sub(expr1,expr2) ->
    let e1, v1 = scope_expr_detail env expr1
in
    let e2, v2 = scope_expr_detail v1 expr2 in
    Sub(e1, e2), env
 | Ast.Mult(expr1,expr2) ->
    let e1, v1 = scope_expr_detail env expr1
in
    let e2, v2 = scope_expr_detail v1 expr2 in
    Mult(e1, e2), v2
 | Ast.Div(expr1,expr2) ->
    let e1, v1 = scope_expr_detail env expr1
in
    let e2, v2 = scope_expr_detail v1 expr2 in
    Div(e1, e2), env
 | Ast.Expo(expr1,expr2) ->
    let e1, v1 = scope_expr_detail env expr1
in
    let e2, v2 = scope_expr_detail v1 expr2 in
    Expo(e1, e2), env
 | Ast.Mod(expr1,expr2) ->
    let e1, v1 = scope_expr_detail env expr1
in
    let e2, v2 = scope_expr_detail v1 expr2 in
    Mod(e1, e2), env
 | Ast.And(expr1,expr2) ->
    let e1, v1 = scope_expr_detail env expr1
in
    let e2, v2 = scope_expr_detail v1 expr2 in
    And(e1, e2), env
 | Ast.Or(expr1,expr2) ->
    let e1, v1 = scope_expr_detail env expr1
in
    let e2, v2 = scope_expr_detail v1 expr2 in
    Or(e1, e2), env
```

```
 | Ast.Not(expr1) ->
    let e1, v1 = scope_expr_detail env expr1
in
    Not(e1), v1
 | Ast.FuncCall(s, el) ->

    let helper e = fst (scope_expr_detail env
e) in
    FuncCall(s, List.map helper el), env
 | Ast.FormalDef(id,e) ->
    let e1 = scope_expr_detail env e in
    let t = type_match env (fst e1) in
    let new_env = assign_current_scope id t
env in
    FormalDef(id, fst e1, t), new_env



let rec scope_stmt env = function
 | Ast.Expr(expr) -> let e, new_env =
scope_expr_detail env expr in

                Expr(e), new_env
 | Ast.Block(blk) ->

    let rec pass_envs env = function
     | [] -> []
     | [s] -> let (s, new_env) = scope_stmt
env s in [s]
     | hd :: tl ->  let (s, new_env) =
scope_stmt env hd in
                [s]@(pass_envs new_env tl)
in



      let helper henv hstmts = snd (scope_stmt
henv hstmts) in
```

```
      let block_env = List.fold_left helper
env (blk) in


    Block(pass_envs env blk), block_env

 | Ast.If(expr,stmt1,stmt2) -> let i, v =
scope_expr_detail env expr in
                        let block1,
new_env1 = scope_stmt env stmt1 in
                        let block2,
new_env2 = scope_stmt env stmt2 in
                        If(i, block1,
block2), new_env2
 | Ast.For(str,expr2,expr3,stmt) ->
    let new_env = assign_current_scope str Int
env in
    let e2, v2 = scope_expr_detail new_env
expr2
    and e3, v3 = scope_expr_detail new_env
expr3 in
    let r = (scope_stmt new_env stmt) in
    For(str, e2, e3, fst r), snd r
 | Ast.While(expr,stmt) ->

    let compare = scope_expr_detail env expr
in
    let block = scope_stmt env stmt in
    While(fst compare, fst block), snd block
 | Ast.Return(e) ->

    let e1, v1 = scope_expr_detail env e in
    Return(e1), v1



let scope_func env = function
 | Ast.FunctionDef(str, el, stmt) ->
```

```
    (* Initialize formal arguments *)
     let init_formals env1 forms =
       let helper henv hforms =

         snd (scope_expr_detail henv hforms) in
       List.fold_left helper env1 forms in
     let new_env = init_formals env el in
     let block, new_env = scope_stmt new_env
stmt in
     let ret_type = (type_of_stmt new_env
block) in
     let helper2 e = fst (scope_expr_detail
new_env e) in


     (* Add function formals to env *)
     let rec formal_type_list env = function
       | [] -> []
       | hd::tl -> type_match env hd ::
formal_type_list env tl in
     let forms = (List.map helper2 el) in
     let my_formal_type_list = formal_type_list
new_env forms in
     let new_form_map = FuncMap.add str
my_formal_type_list new_env.func_tbl_formals in
     let new_fname_map = FuncMap.add str
ret_type new_env.func_tbl in
     let new_env = reassign_symb_tbl_stk
new_env.symb_tbl_stk new_fname_map new_form_map in
     FunctionDef(str, forms, block), new_env



let run_stmts env stmts =
 let helper henv hstmts = snd (scope_stmt henv
hstmts) in
 List.fold_left helper env stmts
```

```
let run_funcs env funcs =
 let helper henv hfuncs = snd (scope_func henv
hfuncs) in
 List.fold_left helper env funcs




let program program =
 let init_print = FuncMap.add "print"
(type_match init_env Na) init_env.func_tbl in
 let init_env = reassign_symb_tbl_stk
init_env.symb_tbl_stk init_print init_env.func_tbl_formals in



 let funcs_rev = List.rev (fst program) in
 let stmts_rev = List.rev (snd program) in



 let new_env1 = run_funcs init_env funcs_rev in



 (*don't want things in function's scopes to be
able to be accessed outside*)
 let init_env = reassign_symb_tbl_stk
init_env.symb_tbl_stk new_env1.func_tbl new_env1.func_tbl_formals in



 let rec pass_envs env = function
```

```
    | [] -> []
    | [s] -> let s = scope_stmt env s in
[s]
    | hd :: tl ->  let s = scope_stmt env
hd in
                (pass_envs (snd s) tl)@[s]
in



 let helper1 env e = (scope_func env e) in



 (List.map (helper1 init_env) funcs_rev),
pass_envs init_env (stmts_rev)
```

sast.ml

```
open Environment



type op =

 | Add
 | Sub
 | Mult
 | Div
 | Expo
 | Mod
 | Assign
 | And
 | Or
 | Not
```

```
type expr_detail =

 | Id of string
 | NaLit of t

 | IntLit of int
 | IntExpr of expr_detail * t
 | BoolLit of bool
 | FloatLit of float
 | StringLit of string
 | Vector of string * expr_detail list * t
 | VectAcc of string * expr_detail * t
 | Matrix of string * expr_detail list *
expr_detail * expr_detail * t
 | MatrixAcc of string * expr_detail *
expr_detail * t
 | Eq of expr_detail * expr_detail * t
 | Neq of expr_detail * expr_detail * t
 | StrEq of expr_detail * expr_detail * t
 | StrNeq of expr_detail * expr_detail * t
 | Add of expr_detail * expr_detail * t
(*   | MatrixAdd of expr_detail * expr_detail *
t
 | MatrixMult of expr_detail * expr_detail * t
*)
 | Sub of expr_detail * expr_detail  * t
 | Mult of expr_detail * expr_detail * t
 | Div of expr_detail * expr_detail * t
 | Expo of expr_detail * expr_detail * t
 | Mod of expr_detail * expr_detail * t
 | FuncCall of string * expr_detail list * t
 | PrintCall of expr_detail * t
 | PrintMatrixCall of expr_detail * t
 | Assign of string * expr_detail * t
 | Update of string * expr_detail * t
 | And of expr_detail * expr_detail * t
 | Or of expr_detail * expr_detail * t
```

```
 | Not of expr_detail * t
 | FormalDef of string * expr_detail * t *
environment



type detail =

        | ExprDet of expr_detail



type expression =

 | Sexpr of expr_detail * t
 | Sadd of expression * expression * t
 | Ssub of expression * expression * t
 | Smult of expression * expression * t
 | Sexpo of expression * expression * t
 | Smod of expression * expression * t
 | SfuncCall of expression list * t
 | Sand of expression * expression * t
 | Sor of expression * expression * t
 | Snot of expression * t



type statement =

 | Sstmt of expression * t
 | Sblock of statement list * t
 | Sif of expression * statement * statement *
t
 | Sfor of string * expression * expression *
statement * t
 | Swhile of expr_detail * statement * t
 | Sreturn of expression * t
```

```
type function_definition =
 | FunctionDef of string * expr_detail list *
statement * t



let rec type_of_stmt = function
 | Sstmt(e,t) -> t
 | Sblock(sl, t) -> let last_stmt = List.nth
(List.rev sl) (List.length sl - 1) in
                    type_of_stmt last_stmt
 | Sif(e,s1,s2,t) -> t
 | Sfor(s1,e1,e2,s2,t) -> t
 | Swhile(e,s,t) -> t
 | Sreturn(e, t) -> t



let rec expr = function
 | Environment.Id (s), env -> Id(s),
Environment.find_type s env
 | Environment.Assign(id, e), env ->
      let e1 = expr (e, env) in
      Assign(id, fst e1, snd e1), snd e1
 | Environment.Update(id, e), env ->

      let e1 = expr (e, env) in
      Update(id, fst e1, snd e1), snd e1
 | Environment.IntLit(c), env -> IntLit(c),
Int
 | Environment.FloatLit(f), env ->
FloatLit(f), Float
 | Environment.BoolLit(b), env ->
BoolLit(b), Bool
 | Environment.StringLit(s), env ->
StringLit(s), String
```

```
 | Environment.Vector(s, vl), env ->let head
= List.hd vl in
                    let _, vtype = expr
(head, env) in
                    let helper e = fst (expr
(e, env)) in
                    Vector(s, (List.map
helper vl), vtype), Vector
 | Environment.VectAcc(s, e), env ->
      let e1 = expr (e, env) in
      let t = snd e1 in
      if (t = Int) then (
           let v_type = Environment.find_type
s env in

           VectAcc(s,
                    fst e1,
                    v_type), v_type
         )
      else
         failwith "Type incompatibility"
 | Environment.Matrix(s, v, nr, nc), env ->
      let head = List.hd v in
      let _, vtype = expr (head, env) in
      let nrv, nrt = expr (nr, env) in
      let ncv, nct = expr (nc, env) in
      let helper e = fst (expr (e, env)) in
      if (nrt != Int || nct != Int) then
        failwith "nrow and ncol must be
integers"
      else (
         Matrix(s, (List.map helper v),
helper nr , helper nc, vtype), Matrix
        )

 | Environment.MatrixAcc(s, e1, e2), env ->
      let ed1 = expr (e1, env) in
      let ed2 = expr (e2, env) in
      let t1 = snd ed1 in
```

```
      let t2 = snd ed2 in
      if (t1 = Int && t2 = Int) then (
        let m_type = Environment.find_type
(s^"type") env in

        MatrixAcc(s, fst ed1, fst ed2,
m_type), m_type
      )
      else
         failwith "Type incompatibility"
 | Environment.Na, env -> NaLit(Na), Na
         | Environment.FuncCall(id, el), env ->
      (* Check that you're only printing one
thing *)
                  if(id = "print") then (

        let print_length = List.length el in
        if (print_length != 1) then
         failwith "print only takes one
argument"
        else (
         let print_arg, print_arg_type = expr
(List.hd el, env) in



         if(print_arg_type = Matrix) then (
           PrintMatrixCall(print_arg,
print_arg_type), Na
         ) else(
          PrintCall(print_arg,
print_arg_type), Na
         )
        )
      )

      else
```

```
      (*iterate over list of expressions and
pull out the expression_detail from each one*)
      let helper1 e = fst (expr (e, env)) in
      let helper2 e = snd (expr (e, env)) in



      (* compare actuals and formals *)
      let actual_types = List.map helper2 el
in
      let formal_types =
Environment.FuncMap.find id env.func_tbl_formals in
      if (actual_types = formal_types) then (
        let ret_type =
Environment.FuncMap.find id env.func_tbl in
        FuncCall(id, (List.map helper1 el),
ret_type), ret_type )

      else

        failwith "Illegal function
arguments"
         | Environment.Eq( e1, e2), env ->
                let e1 = expr (e1, env)
                and e2 = expr (e2, env) in
                let _, t1 = e1
                and _, t2 = e2 in
                if (t1 == t2) then (
            if t1 == String then
                                    StrEq((fst e1), (fst e2), t1), t1
            else
               Eq((fst e1), (fst e2), t1),
t1
                          )
                else
                          failwith "Type incompatibility"
         | Environment.Neq( e1, e2), env ->
                let e1 = expr (e1, env)
                and e2 = expr (e2, env) in
```

```
                let _, t1 = e1
                and _, t2 = e2 in
                if (t1 == t2 ) then (
            if t1 == String then
                                    StrNeq((fst e1), (fst e2), t1), t1
            else
               Neq((fst e1), (fst e2), t1),
t1
                        )
                else
                        failwith "Type incompatibility"
        | Environment.Add( e1, e2), env ->
                let e1 = expr (e1, env)
                and e2 = expr (e2, env) in
                let _, t1 = e1
                and _, t2 = e2 in
                if (t1 == t2 && (t1 == Int || t1 ==
Float)) then (
                                Add((fst e1), (fst e2), t1), t1
                        )
                else if ( t1 == t2 && t1 == Matrix)
then (
            Add((fst e1), (fst e2), t1),
Matrix
          )
      else
         failwith "Type incompatibility"
 | Environment.Sub( e1, e2 ), env ->
        let e1 = expr (e1, env)
        and e2 = expr (e2, env) in
        let _, t1 = e1
        and _, t2 = e2

      in
                if (t1 == t2 && (t1 == Int || t1 ==
Float)) then (
           Sub((fst e1),(fst e2), t1), t1
           )
        else
```

```
            failwith "Type
incompatability"
 | Environment.Mult( e1, e2 ), env ->
        let e1 = expr (e1, env)
        and e2 = expr (e2, env) in
        let _, t1 = e1
        and _, t2 = e2 in
        if (t1 == t2 && (t1 == Int ||
t1 == Float)) then (
            Mult((fst e1), (fst e2), t1), t1
          )
        else if ( t1 == t2 && t1 ==
Matrix) then (
              Mult((fst e1), (fst e2), t1),
Matrix
          )
        else
            failwith "Type
incompatibility"
 | Environment.Div( e1, e2 ), env ->
        let e1 = expr (e1, env)
        and e2 = expr (e2, env) in
        let _, t1 = e1
        and _, t2 = e2 in
    if (t1 == t2 && (t1 == Int || t1
== Float)) then (
            Div((fst e1),(fst e2), t1), t1
            )
        else
            failwith "Type
incompatability"
 | Environment.Expo( e1, e2 ), env ->
        let e1 = expr (e1, env)
        and e2 = expr (e2, env) in
        let _, t1 = e1
        and _, t2 = e2 in
          if ((t1 == Int || t1 == Float) && (t2
== Int)) then (
            Expo((fst e1),(fst e2), t1), t1
```

```
            )
        else
            failwith "Type
incompatability"
 | Environment.Mod( e1, e2 ), env ->
        let e1 = expr (e1, env)
        and e2 = expr (e2, env) in
        let _, t1 = e1
        and _, t2 = e2 in
          if (t1 == Int || t1 == Float && t2 ==
Int) then (
            Mod((fst e1),(fst e2), t1), Int
            )
        else
            failwith "Type
incompatability"
 | Environment.And( b1, b2), env ->
        let b1 = expr (b1, env)
        and b2 = expr (b2, env) in
        let _, t1 = b1
        and _, t2 = b2 in
          if (t1 == t2 && (t1 == Bool)) then (
              And((fst b1),(fst b2), Bool),
Bool
            )
        else
            failwith "Type
incompatibility"
 | Environment.Or( b1, b2), env ->
        let b1 = expr (b1, env)
        and b2 = expr (b2, env) in
        let _, t1 = b1
        and _, t2 = b2 in
          if (t1 == t2 && (t1 == Bool)) then (
              Or((fst b1),(fst b2), Bool),
Bool
            )
        else
```

```
            failwith "Type
incompatibility"
 | Environment.Not( b1 ), env ->
        let b1 = expr (b1, env) in
        let _, t1 = b1 in
        if ( t1 == Bool) then (
              Not((fst b1), Bool), Bool
           )
        else
           failwith "Type
incompatibility"
 | Environment.FormalDef(id,e,t), env ->

        let e1 = expr (e, env) in
        let _, t1 = e1 in
        FormalDef(id, fst e1, t1, env), t1
let rec stmt = function
          | Environment.Expr( e ), env ->

      let r = expr (e, env) in
              Sstmt(Sexpr(fst r, snd r), (snd r))
 | Environment.Block( sl ), env ->

        let rec pass_envs env = function
              | hd :: tl ->  let s = stmt
(hd, env) in
                             let passed =
pass_envs env (tl) in
                             (passed)@[s]
              | [] -> [] in
        let l = pass_envs env sl in
        let last_stmt = List.nth l
(List.length l - 1) in
        Sblock(l, type_of_stmt last_stmt)

 | Environment.If(e, s1, s2), env ->

        let r = expr (e, env) in
        let stmt1 =  stmt (s1, env) in
```

```
        let stmt2 =  stmt (s2, env) in
        let t1 = type_of_stmt stmt1 in

        let t2 = type_of_stmt stmt2 in
        if(t1 = t2) then (
          Sif(Sexpr( (fst r), (snd r) ),

              stmt1, stmt2, t1)
        )else
          failwith "Type incompatibility
in if else statement"
 | Environment.For(s, ie1, ie2, b), env ->

        let rie1 = expr (ie1, env)
        and rie2 = expr (ie2, env) in
        let b1 = stmt (b, env) in
        let t = type_of_stmt b1 in
        Sfor(s,

            Sexpr(fst(rie1),snd(rie1)),

            Sexpr(fst(rie2),snd(rie2)),

            b1,
            t)
 | Environment.While(e, s), env ->

        let e_val = expr (e, env) in
        let s_val = stmt (s, env) in
        Swhile(fst e_val, s_val, Na)
 | Environment.Return(e), env ->

        let e1 = expr (e, env) in
        Sreturn(Sexpr(fst e1, snd e1), snd e1)



let func_def = function
 | Environment.FunctionDef(s, ids, b), env ->
```

```
        let helper2 e = fst (expr (e, env)) in
        let forms = List.map helper2 ids in
        let block = stmt (b, env) in
        FunctionDef(s, forms, block,
type_of_stmt block)



let program program =

         List.map func_def (fst program), List.map stmt
(snd program)
```

cast.ml

```
open Environment



type op =

 | Eq
 | Neq
 | Add
 | Sub
 | Mult
 | Div
 | Expo
 | Mod
 | Assign
 | And
 | Or
 | Not
```

```
type ct =

 | String
 | Int
 | Float
 | Bool
 | Void
 | Vector
 | Matrix
 | IdType



type cexpr_detail =

  | Na of string * ct
  | Id of string * ct
  | IntLit of int
  | IntExpr of cexpr_detail * ct
  | FloatLit of float
  | BoolLit of bool
  | StringLit of string
  | Vector of string * cexpr_detail list * ct
  | VectAcc of string * cexpr_detail * ct
  | Matrix of string * cexpr_detail list *
cexpr_detail * cexpr_detail * ct
  | MatrixAcc of string * cexpr_detail *
cexpr_detail * ct
  | Eq of cexpr_detail * cexpr_detail * ct
  | Neq of cexpr_detail * cexpr_detail * ct
  | StrEq of cexpr_detail * cexpr_detail * ct
  | StrNeq of cexpr_detail * cexpr_detail * ct
  | Add of cexpr_detail * cexpr_detail * ct
  | Sub of cexpr_detail * cexpr_detail * ct
  | Mult of cexpr_detail * cexpr_detail * ct
  | Div of cexpr_detail * cexpr_detail * ct
  | Expo of cexpr_detail * cexpr_detail * ct
```

```
  | Mod of cexpr_detail * cexpr_detail * ct
  | FuncCall of string * cexpr_detail list * ct
  | PrintCall of cexpr_detail * ct
  | PrintMatrixCall of cexpr_detail * ct
  | Assign of string * cexpr_detail * ct
  | Update of string * cexpr_detail * ct
  | And of cexpr_detail * cexpr_detail * ct
  | Or of cexpr_detail * cexpr_detail * ct
  | Not of cexpr_detail * ct
  | FormalDef of string * cexpr_detail * ct



type cexpression =

 | Cexpr of cexpr_detail * ct
 | Ceq of cexpression * cexpression * ct
 | Cneq of cexpression * cexpression * ct
 | Cadd of cexpression * cexpression * ct
 | Csub of cexpression * cexpression * ct
 | Cmult of cexpression * cexpression * ct
 | Cdiv of cexpression * cexpression * ct
 | CFAdd of cexpression * cexpression * ct
 | CFSub of cexpression * cexpression * ct
 | CFMult of cexpression * cexpression * ct
 | CFDiv of cexpression * cexpression * ct
 | CMatrixAcc of cexpression * cexpression * ct
 | Cexpo of cexpression * cexpression * ct
 | Cmod of cexpression * cexpression * ct
 | CfuncCall of cexpression list * ct
 | Cand of cexpression * cexpression * ct
 | Cor of cexpression * cexpression * ct
 | Cnot of cexpression * ct



type statement =

 | Cstmt of cexpression * ct
```

```
 | Cblock of statement list * ct
 | Cif of cexpression * statement * statement *
ct
 | Cfor of string * cexpression * cexpression *
statement * ct
 | Cwhile of cexpr_detail * statement * ct
 | Creturn of cexpression * ct



type func_decl_detail = {
   fname : string;
   formals : cexpr_detail list;
   body : statement;
}



type func_decl =

 | CFunctionDef of func_decl_detail * ct



type program =

 func_decl list



let rec string_of_matrix_assign = function
 | Matrix(s, el, e, e1, ct) -> s
 | Assign(s, e, ct) ->
string_of_matrix_assign e
 | Id(s, ct) -> s
 | _ -> failwith "matrix required"
```

```
let rec string_of_ctype = function
 | String -> "string"
 | Float -> "float"
 | Int -> "int"
 | Bool -> "bool"
 | Void -> "void"
 | IdType -> "IdType"
 | Vector -> "Vector"
 | Matrix -> "vector<vector<int>
>"



let rec type_match = function
 | Environment.String -> String
 | Environment.Int -> Int
 | Environment.Float -> Float
 | Environment.Bool -> Bool
 | Environment.Na -> Void
 | Environment.Vector -> Vector
 | Environment.Matrix -> Matrix



let rec cexpr_detail = function
| Sast.Id(s) ->  Id(s, String)
| Sast.IntLit(i) -> IntLit(i)
| Sast.FloatLit(i) -> FloatLit(i)
| Sast.BoolLit(b) -> BoolLit(b)
| Sast.StringLit(s) -> StringLit(s)
| Sast.IntExpr(e,t) -> IntExpr(cexpr_detail
e, type_match t)
| Sast.Vector(s, v, t) -> let ct =
type_match t in
       Vector(s, List.map cexpr_detail v, ct)
| Sast.VectAcc(s, e1, t) ->
      let cexp1 = cexpr_detail e1 in
      VectAcc(s, cexp1, type_match t)
```

```
| Sast.NaLit(t) -> Na("Void",
type_match t)
| Sast.Matrix(s, v, nr, nc, t) ->

                        let ct =
type_match t in
                        Matrix(s,
List.map cexpr_detail v, cexpr_detail nr, cexpr_detail nc, ct)
| Sast.MatrixAcc(s, e1, e2, t) ->
      let cexp1 = cexpr_detail e1 in
       let cexp2 = cexpr_detail e2 in
      MatrixAcc(s, cexp1, cexp2, type_match t)
| Sast.Neq(e1, e2, t) -> Neq((cexpr_detail
e1), (cexpr_detail e2), type_match t)
| Sast.Eq(e1, e2, t) -> Eq((cexpr_detail
e1), (cexpr_detail e2), type_match t)
| Sast.StrEq(e1, e2, t) ->
StrEq((cexpr_detail e1), (cexpr_detail e2), type_match t)
| Sast.StrNeq(e1, e2, t) ->
StrNeq((cexpr_detail e1), (cexpr_detail e2), type_match t)
| Sast.Add(e1, e2, t) -> Add((cexpr_detail
e1), (cexpr_detail e2), type_match t)
| Sast.Sub(e1, e2, t) -> Sub(cexpr_detail
e1, cexpr_detail e2, type_match t)
| Sast.Mult(e1, e2, t) -> Mult(cexpr_detail
e1, cexpr_detail e2, type_match t)
| Sast.Div(e1, e2, t) -> Div(cexpr_detail
e1, cexpr_detail e2, type_match t)
| Sast.Expo(e1, e2, t) -> Expo(cexpr_detail
e1, cexpr_detail e2, Int)
| Sast.Mod(e1, e2, t) -> Mod(cexpr_detail
e1, cexpr_detail e2, Int)
| Sast.FuncCall(id, el, t) -> let ct =
type_match t in
                        FuncCall(id,
List.map cexpr_detail el, ct)
| Sast.PrintCall(e, t) -> let ct =
type_match t in
```

```
PrintCall(cexpr_detail e, ct)
| Sast.PrintMatrixCall(e, t) -> let ct =
type_match t in

PrintMatrixCall(cexpr_detail e, ct)
| Sast.Assign(id, e, t) -> let ct =
type_match t in
                        Assign(id,
cexpr_detail e, ct)
| Sast.Update(id, e, t) -> let ct =
type_match t in
                         Update(id,
cexpr_detail e, ct)
| Sast.And(e1, e, t) -> let ct = type_match
t in

                       And(cexpr_detail e1,
cexpr_detail e, ct)
| Sast.Or(e1, e2, t) ->  let ct = type_match
t in
                       Or(cexpr_detail e1,
cexpr_detail e2, ct)
| Sast.Not(e, t) -> let ct = type_match t in
                   Not(cexpr_detail e, ct)
| Sast.FormalDef(id, e, t, env) ->

    let t = Environment.find_type id env in
    FormalDef(id, cexpr_detail e, type_match
t)


let rec cexpr = function
 | Sast.Sexpr(e, t) -> Cexpr(cexpr_detail e,
type_match t)
 | Sast.Sadd(e1, e2, t) -> Cadd(cexpr e1,
cexpr e2, type_match t)
 | Sast.Ssub(e1, e2, t) -> Csub(cexpr e1,
cexpr e2, type_match t)
```

```
 | Sast.Smult(e1, e2, t) -> Cmult(cexpr e1,
cexpr e2, type_match t)
 | Sast.Sexpo(e1, e2, t) -> Cexpo(cexpr e1,
cexpr e2, type_match t)
 | Sast.Smod(e1, e2, t) -> Cmod(cexpr e1,
cexpr e2, type_match t)
 | Sast.SfuncCall(el, t) ->
CfuncCall((List.map cexpr el), type_match t)
 | Sast.Sand(e1, e2, t) -> Cand(cexpr e1,
cexpr e2, type_match t)
 | Sast.Sor(e1, e2, t) -> Cor(cexpr e1,
cexpr e2, type_match t)
 | Sast.Snot(e, t) -> Cnot(cexpr e,
type_match t)



let rec stmt = function
 | Sast.Sstmt(e, t) -> let r = cexpr e in
                       Cstmt(r, type_match t)
 | Sast.Sblock(sl, t) -> let stmt_rev = sl
in
                       let l = List.map stmt
stmt_rev in
                       Cblock(List.rev l,
type_match t)
 | Sast.Sif(e, s1, s2, t) -> let r = cexpr e
in
                            Cif(r, stmt s1,
stmt s2, type_match t)
 | Sast.Sfor(id, e1, e2, s, t) -> Cfor(id,
cexpr e1, cexpr e2, stmt s, Void)
 | Sast.Swhile(e, s, t) ->
Cwhile(cexpr_detail e, stmt s , type_match t)
 | Sast.Sreturn(e, t) -> let r = cexpr e in
                        Creturn(r, type_match
t)
```

```
let rec add_return = function
 | Cblock(sl, t) ->

                    let last_stmt =
List.nth sl (List.length sl - 1) in
                    let sl = List.tl
(List.rev sl) in
                    let sl = List.rev sl
in
                    let return = match
last_stmt with
                       | Cstmt(e,t) ->
Creturn(e, t)

                       | Cblock(sl, t) ->
let last_stmt = List.nth sl (List.length sl - 1) in

add_return last_stmt
                       | Cif(e, s1, s2,
t) -> Cif(e, add_return s1, add_return s2, t)
                       | Cfor(s1, e1, e2,
s2, t) -> add_return s2
                       | Cwhile(e,s,t) ->
Cwhile(e, add_return s ,t)
                       | Creturn(e, t) ->
Creturn(e, t)
                    in
                    Cblock(sl@[return], t)
 | _ -> failwith "Unable to identify a
block"



let func_def = function
 | Sast.FunctionDef(s, frmls, b, t) -> let
block =
```

```
if(type_match t != Void) then

add_return (stmt b)
                                    else
                                      stmt
b in
                                   let
func_det =


    {

      fname = s;

      formals = List.map cexpr_detail frmls;

      body = block

    } in

CFunctionDef(func_det, type_match t)
let program sast =

 let functions = List.map func_def (fst sast)
in
 let main = CFunctionDef({
              fname = "main";
              formals = [];
              body = Cblock(List.rev
(List.map stmt (snd sast)), Void)
           }, Int) in
 functions@[main]
```

acc.ml

```
open Parser
open Scanner
open Ast
open Sast
open Environment
open Cast




let _ =
 let lexbuf = Lexing.from_channel stdin in
 let program = Parser.program Scanner.token
lexbuf in
 let envr = Environment.program program in
 let sast = Sast.program envr in
 let cast = Cast.program sast in




let rec compile_detail = function
 | Cast.Na(s, t) -> s
 | Cast.Id(s, t) -> s
 | Cast.IntLit(i) -> string_of_int i
 | Cast.IntExpr(e, t) -> compile_detail e
 | Cast.FloatLit(f) -> string_of_float f
 | Cast.BoolLit(b) -> string_of_bool b
 | Cast.StringLit(s) -> " \""
^ s ^ "\" "
 | Cast.Vector(s, v, t) ->

        let helper e = compile_detail e in
        let holder = s ^ "holder" in
        let ty = Cast.string_of_ctype t in
        ty ^ " " ^ holder ^ "[]
= {" ^
```

```
        (String.concat ", "
(List.map helper v)) ^ "};\n" ^
        "vector<" ^ ty ^ ">
" ^ s ^

        " (" ^ holder ^ ", "
^ holder ^ " + sizeof(" ^

        holder ^ ") / sizeof(" ^ ty
^"))"
 | Cast.VectAcc(e1, e2, t) ->
        e1 ^ "[" ^ compile_detail e2
^ "-1]"
 | Cast.Matrix(s, v, nr, nc, t) ->
      let helper e = compile_detail e in
        let holder = s ^ "Holder" in
        let count = s ^ "count" in
        let nr_str = compile_detail nr in
        let nc_str = compile_detail nc in

        let matrix = s in

        let ty = Cast.string_of_ctype t in
        ty ^ " " ^ holder ^ "[]
= {" ^

        (String.concat ", "
(List.map helper v)) ^ "};\n" ^
        "vector<" ^ ty ^ ">
" ^ matrix ^ "Vector" ^
        " (" ^ holder ^ ", "
^ holder ^ " + " ^
        nc_str ^ " / sizeof(" ^ ty ^
"));\n" ^
        "vector<vector<" ^ ty
^ "> > " ^ matrix ^ " (" ^ nr_str ^ ");\n"
^
        "int " ^ count ^ "=0;\n"
^
```

```
        "for(int i=0; i<" ^
nr_str ^ "; i++) { \n" ^
        matrix ^ "[i].resize(" ^
nc_str ^ ");\n" ^
        "for(int j=0; j<" ^
nc_str ^ "; j++) { \n" ^
        matrix ^ "[i][j] = " ^
holder ^ "[" ^ count ^ "++];\n}\n}"
 | Cast.MatrixAcc(m, r, c, t) ->
        m ^ "[" ^ compile_detail r ^
"][" ^ compile_detail c ^ "]"
 | Cast.FuncCall(id, el, t) -> let helper e
= compile_detail e in
                              id ^ "("
^ (String.concat ", " (List.map helper el)) ^ ")"
 | Cast.PrintCall(e, t) -> "cout <<
" ^ (compile_detail e) ^ ";\n" ^

                          "cout <<
endl"
 | Cast.PrintMatrixCall(e, t) ->
"print_matrix("^ (compile_detail e) ^")"


 | Cast.And(b1, b2, t) -> (compile_detail
b1) ^ " && " ^

                           (compile_detail b2)
 | Cast.Or(b1, b2, t) -> (compile_detail b1)
^ " || " ^

                         (compile_detail b2)
 | Cast.Not(b1, t) -> "! " ^
(compile_detail b1)
 | Cast.Update(id, e, t) -> id ^ " = "
^ (compile_detail e)
 | Cast.Assign(id, e, t) ->

        if(t = Matrix) then (
```

```
            "vector<vector<int>
> " ^id^ " = " ^ (compile_detail e)
         ) else(
           let ty = (string_of_ctype t) in
           ty ^ " " ^ id ^ " = "
^ (compile_detail e)
         )
 | Cast.Mod(e1, e2, t) -> (compile_detail
e1) ^ " % " ^

                        (compile_detail e2)

 | Cast.Expo(e1, e2, t) -> "pow("
^ (compile_detail e1) ^ ",  " ^

                         (compile_detail e2)
^ ")"
 | Cast.Eq(e1, e2, t) -> "(" ^
(compile_detail e1) ^ " == " ^
                        (compile_detail e2) ^
")"
 | Cast.Neq(e1, e2, t) -> "(" ^
(compile_detail e1) ^ " != " ^
                        (compile_detail e2) ^
")"
 | Cast.StrEq(e1, e2, t) -> "(strcmp("
^

                        (compile_detail e1) ^
"," ^
                        (compile_detail e2) ^
") == 0)"
 | Cast.StrNeq(e1, e2, t) -> "(strcmp("
^

                        (compile_detail e1) ^
"," ^
                        (compile_detail e2) ^
") != 0)"
```

```
 | Cast.Div(e1, e2, t) -> (compile_detail
e1) ^ " / " ^

                        (compile_detail e2)
 | Cast.Mult(e1, e2, t) -> if(t = Matrix)
then (
                          let mid1 =
string_of_matrix_assign e1 in
                         let mid2 =
string_of_matrix_assign e2 in

"matrix_mult("^mid1^", "^ mid2 ^ ")"
                        )else(
                           (compile_detail
e1) ^ " + " ^
                           (compile_detail
e2)
                         )
 | Cast.Sub(e1, e2, t) -> (compile_detail
e1) ^ " - " ^

                        (compile_detail e2)

 | Cast.Add(e1, e2, t) -> if(t = Matrix)
then (
                          let mid1 =
string_of_matrix_assign e1 in
                         let mid2 =
string_of_matrix_assign e2 in

"matrix_add("^mid1^", "^ mid2 ^ ")"
                        )else(
                           (compile_detail
e1) ^ " + " ^
                           (compile_detail
e2)
                         )
```

```
 | Cast.FormalDef(id, e, t) ->
Cast.string_of_ctype t ^ " " ^ id ^ "=" ^
(compile_detail e)



 in



let rec compile_expr = function
        | Cexpr(e, t) -> compile_detail e

        | CfuncCall(el, t) -> String.concat ""
(List.map compile_expr el)
 | Cmod(e1, e2, t) -> compile_expr e1 ^
compile_expr e2

 | Cexpo(e1, e2, t) -> compile_expr e1 ^
compile_expr e2
 | Cdiv(e1, e2, t) -> compile_expr e1 ^
compile_expr e2

 | Cmult(e1, e2, t) -> compile_expr e1 ^
compile_expr e2
 | Csub(e1, e2, t) -> compile_expr e1 ^
compile_expr e2

 | Cadd(e1, e2, t) -> compile_expr e1 ^
compile_expr e2

 | CFAdd(e1, e2, t) -> compile_expr e1 ^
compile_expr e2

 | CFSub(e1, e2, t) -> compile_expr e1 ^
compile_expr e2
```

```
 | CFMult(e1, e2, t) -> compile_expr e1 ^
compile_expr e2

 | CMatrixAcc(e1, e2, t) -> compile_expr e1
^ compile_expr e2
 | Ceq(e1, e2, t) -> compile_expr e1 ^
compile_expr e2

 | Cneq(e1, e2, t) -> compile_expr e1 ^
compile_expr e2

 | CFDiv(e1, e2, t) -> compile_expr e1 ^
compile_expr e2

 | Cand(b1, b2, t) -> compile_expr b1 ^
compile_expr b2

 | Cor(b1, b2, t) -> compile_expr b1 ^
compile_expr b2
 | Cnot(b1, t) -> (compile_expr b1) in



let rec compile_stmt = function
 | Cstmt(e, t) -> compile_expr e ^ ";\n"
 | Cblock(sl, t) -> let string_list l =
List.map compile_stmt l in
                "{\n" ^
String.concat "" (string_list sl) ^ "}\n"
 | Cif(e, s1, s2, t) ->   "if(" ^
compile_expr e ^ ")"

                        ^ (compile_stmt s1)
^

                        "else"

                        ^  (compile_stmt s2)
```

```
 | Cfor(id, e1, e2, s, t) -> "for( int
" ^ id ^ "="  ^ compile_expr e1 ^ "; " ^

                        id ^" <= "
^ compile_expr e2 ^ "; " ^ id ^
                        "++)\n" ^
compile_stmt s

 | Creturn(e, t) -> "return (" ^
(compile_expr e) ^ ");\n"
 | Cwhile(e, s, t) -> "while " ^
(compile_detail e) ^ compile_stmt s

 in



let compile_func_detail f =

 let helper e = compile_detail e in
 let string_formals = List.map helper f.formals
in
   f.fname
   ^

   "(" ^ String.concat ","
string_formals ^ ")"
   ^

   compile_stmt f.body
   in



let compile_func = function

   | CFunctionDef(f, t) ->

       if(t = Matrix) then (
```

```
        string_of_ctype t ^ " " ^
        compile_func_detail f
      )else(
        string_of_ctype t ^ " " ^
        compile_func_detail f
      ) in


let compile cast =

 let string_list = List.map compile_func cast
in



 String.concat ""  string_list in



 let c_begin =

  "#include<iostream>\n"^
  "#include<stdio.h>\n" ^
  "#include<math.h>\n" ^
  "#include<vector>\n" ^
  "#include<string>\n" ^
  "#include<string.h>\n" ^
  "using namespace std;\n" ^
  "
  vector<vector<int> >
matrix_add(vector<vector<int> > a, vector<vector<int>
> b){
     vector<vector<int> > c;
     for(int i = 0; i < a.size(); i++){
        vector<int> row;
         for(int j = 0; j < a[0].size();
j++){
             row.push_back(a[i][j] +
b[i][j]);
```

```
        }
        c.push_back(row);
    }
    return c;
 }



 vector<vector<int> >
matrix_mult(vector<vector<int> > a, vector<vector<int>
> b){
 vector<vector<int> > c;
 int i,j,k;
 #pragma omp parallel shared(a,b,c)
private(i,j,k)
 {
 #pragma omp for  schedule(static)
   for (i=0; i<a.size(); i=i+1){
            vector<int> row;
            for (j=0; j<b[0].size();
j=j+1){
               row.push_back(0);
               for (k=0; k<b.size();
k=k+1){
                  row[j] +=
((a[i][k])*(b[k][j]));
               }
            }
            c.push_back(row);
        }
    }
    return c;
 }
 void print_matrix(vector<vector<int>
> a){
    for(int i = 0; i<a.size(); i++){
        cout<<endl;
        for(int j = 0; j<a[0].size();
j++){
```

```
            cout << a[i][j] << \"
\";
        }
    }
  }\n" in



 print_endline ( c_begin ^ (compile cast))
```

8.2) Test Files

./addTest.acc
```
print(1 + 2)
```

./modTest.acc
```
print( 4 %% 3 )
```

./eqNeqTest.acc
```
print( 1 == 1 )
print( 1 != 1 )
print( 1 == 2 )
print( 1 != 2 )
print( 1.1 == 1.1 )
print( 1.1 != 1.1 )
print( 1.1 == 1.2 )
print( 1.1 != 1.2 )
print( "abc" == "abc" )
print( "abc" != "abc" )
print( "abc" == "def" )
print( "abc" != "def" )
```

./falseLitTest.acc
```
print(false)
```

./floatLitTest.acc

```
print(1.234)

./matrixTest.acc
m <- matrix(c(1,2,3,4,5,6), nrow=2, ncol=3)
for(i in 0:1) {
	print(100)
	for(j in 0:2) {
		print(m[i][j])
	}
}

./whileTest.acc
a<-3
while(a==3) {
	print(a)
	print("hello while")
	a<-4
}
print(a)

./floatOpTest.acc
print(1.23 + 4.56)
print(1.23 + 4.56 + 7.89)
print(10.0 - 4.32)
print(10.0 - 1.23 - 1.40)
print(25.0 / 5.0)
print(125.0 / 5.0 / 5.0)
print(5.0 * 5.0)
print(5.0 * 5.0 * 5.0)

./forTest.acc
for(i in 1:5){
print(i)
}

./divTest.acc
print( 8 / 2 )

./assignAndPrintTest.acc
```

```
a <- 3
print(a)

./stringTest.acc
a <- "Hello World!"
print(a)

./trueLitTest.acc
print(true)

./expoTest.acc
print( 2 ^ 4 )

./subTest.acc
print(4 - 1)

./funcTest.acc
foo <- function(i=0, j=0.0){
        a <- i
        print(a)
        return(3)
}
bar <- function(i=0) {
        i<-3
        i<-5
        return(i)
}
baz <- function() {
        print("hello")
        return("world")
}
a <- foo(4, 1.1)
print(a)
b <- bar(100)
print(b)
c <- baz()
print(c)

./vectorTest.acc
```

```
a <- c(1, 2, 3, 4)
for(i in 1:4){
  print(a[i])
}
b <- c(true, false, false, true)
for(i in 1:4){
  print(b[i])
}
c <- c(1.2, 3.4, 5.6, 7.8)
for(i in 1:4){
  print(c[i])
}
```

./scopeTest.acc

```
if (false) {
a <- 5
print(a)
}
else {
a <- 6
print(a)
}
```

./helloWorldTest.acc

```
foo <- function(i="goodbye",j="moon")
{
        print(i)
        print(j)
        return(5)
}
i <- foo("hello", "world")
print(i)
```

./matrixOperationsTest.acc

```
m <- matrix(c(1,2,3,4,5,6), nrow=2, ncol=3)
for(i in 0:1) {
        print(100)
```

```
	for(j in 0:2) {
		print(m[i][j])
	}
}

./gcdTest.acc
gcd <- function(a=0,b=0, t=0) {
while(b != 0) {
t <- b
b <- a %% b
a <- t
}
return(a)
}
d <- gcd(15,10,0)
print(d)

./matrixMultTest.acc
f <- function(){
a <- matrix(c(1,2,3,4), nrow=2,ncol=2)
b <- matrix(c(4,3,2,1), nrow=2,ncol=2)
a * b
}
print(f())

./assignTest.acc
a <- 1

./andTest.acc
print(true && true)

./matrixAdd.acc
f <- function(){
a <- matrix(c(1,1,1,2,2,2,3,3,3),
nrow=3,ncol=3)
b <- matrix(c(1,1,1,2,2,2,3,3,3),
nrow=3,ncol=3)
a + b
}
```

```
print(f())

./notTest.acc
print(!true)

./ifElseTest.acc
if(true){
	print(true)
}
else{
	print(false)
}

./multTest.acc
print( 2 * 2 )

./multiStatementTest.acc
print(1+2)
print(true)
print(3+4)
./tests.txt
```

8.3) Expected Output

./vectorTestExpected.txt

```
1
2
3
4
1
0
0
1
1.2
3.4
```

5.6
7.8

./trueLitTestExpected.txt
1

./helloWorldTestExpected.txt
hello
world
5

./assignTestExpected.txt

./assignAndPrintTestExpected.txt
3

./whileTestExpected.txt
3
hello while
4

./multiStatementExpected.txt
3
1
7

./matrixAddExpected.txt

2 2 2

4 4 4

6 6 6
./subTestExpected.txt
3

./eqNeqTestExpected.txt

1
0
0
1
1
0
0
1
1
0
0
1

./orTestExpected.txt
1

./matrixTestExpected.txt
100
1
2
3
100
4
5
6

./multTestExpected.txt
4

./matrixMultTestExpected.txt

8 5

20 13

./funcTestExpected.txt
4

3
5
hello
world

./ifElseTestExpected.txt
1

./notTestExpected.txt
0

./divTestExpected.txt
4

./expoTestExpected.txt
16

./falseLitTestExpected.txt
0

./scopeTestExpected.txt
6

./floatOpTestExpected.txt
5.79
13.68
5.68
7.37
5
5
10
15

./modTestExpected.txt
1

./forTestExpected.txt
1
2

3
4
5

./floatLitTestExpected.txt
1.234

./andTestExpected.txt
1

./stringTestExpected.txt
Hello World!

./gcdTestExpected.txt
5

./addTestExpected.txt
3

./multiStatementTestExpected.txt
3
1
7