Programming Languages and Translators
COMS W4115
Spring 2021
Final Report

Amina Assal, Ivan Barral, Rafail Khalilov, and Myric Lehner

Table of Contents

1. Introduction

1.1 Background

NumPy has been a good friend of mathematicians for many years – when dealing with matrices, and especially when using Python to code. The team behind Red Pandas has a lot of respect for math, especially for linear algebra. After having gone through numerous maths and computer-science courses at our university, we found that there is a lot more to be contributed to the list of tools that are currently available to programmers. We decided to design and build a language that recreates some of NumPy's core functionality natively so that your code is cleaner and faster.

1.2 Summary of Language Features

Red Pandas is a strongly typed, imperative programming language built to quickly churn through linear algebra problems. It is statically typed and statically scoped. Moreover, it follows similar operator precedence rules as those of the C language, and assumes most of C's basic operations. By providing a clean syntax, Red Pandas native functionality enables users to write code that solves systems of equations and manipulates data in matrices. Users are able to define variables using identifiers and define blocks of code using both function and block scoping. In addition, user-defined functions are passable into core operations and functions, allowing for a robust library to be developed.

The following sections cover Red Pandas' lexical conventions, syntax, scoping, type conversions, built-in functions, as well as the language's grammar. Examples of code written in Red Pandas are supplemented throughout the language reference manual for reference, with comments included. Also included is a short tutorial on how to program in Red Pandas to get you started quickly.

2. Language Tutorial

2.1 Red Pandas Quickstart

2.1.1 Setting up the Environment

Red Pandas was built and tested using Ubuntu on a Docker image that pre-installs the necessary versions of OCaml and LLVM. To set up your own version of this Docker image, extract it from the MicroC tar file here on Professor Edwards's website.

Once you have moved the file to your Red Pandas directory (and installed Docker), run:

```
docker run --rm -it -v `pwd`:/home/Red-Pandas -w=/home/Red-Pandas columbiasedwards/plt
```

2.1.2 Build the compiler

To build the compiler, extract the Red Pandas folder from the tar.gz file and go into that directory from the command line. (This is also where you put the docker image.) From there, run the above command for Docker and then run `make all`. To test that it has built correctly, next run `make test` to run all the tests in the /Tests folder and see that they all return OK.

2.2 Basic Syntax

2.2.1 Introduction

For the most part, Red Pandas uses syntax that would be familiar to users of C and Java. Variables must be declared at the top of whatever scope they appear in, and scoping is similar to those languages as well: Variables declared at the top, outside of main() or user defined functions are global, other variables are local to the code block (enclosed by curly brackets) in which they are declared. There is no `static` keyword. The only major difference is the matrix type – those can be thought of as two dimensional arrays, but with extra available functionality that might be useful for matrix arithmetic.

2.2.2 Data Types

In addition to integer and floating point number matrices, Red Pandas also comes with a basic String data type that can be printed with the command `printStr()`. Numbers (and values from matrices) can be printed with the `print()` command.

The matrix data type comes with easy manipulation and access to indexed components, as well as a built-in transpose capability. Matrices must be declared with their size before being assigned values, as in the example code below.

2.3 Usage

To compile and run your program written in Red Pandas, use the following command:

```
./rpc <filename>
```

2.4 Example Programs

```
def void printThreeBy(matrix int [3][3] m) {
    int i;
    int j;

    for(i = 0; i < m.row; i = i + 1) {
        for(j = 0; j < m.col; j = j + 1) {
            print(m[i][j]);
        }
        printStr("");
    }
}

def void main() {

    matrix int [3][3] mat;
    matrix int [3][3] alt;

    mat = [[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]];

    printThreeBy(mat);

    printStr("And now a math test:");

    alt = mat.T;

    printThreeBy(mat * alt);
}
```

3. Language Manual

3.1 Lexical Conventions

3.1.1 Comments

The characters /* introduce a comment, which terminates with the characters */.
Single line comments are also available to the programmer, starting with the characters //.

3.1.2 Identifiers (Names)

An identifier is a sequence of letters and digits. The identifier starts with a letter or underscore and could be followed by letters, numbers, or underscores. These identifiers are case sensitive.

```
foo;           /* correct */
fo0;           /* correct */
0of;           /* not a proper identifier*/
f_o;           /* correct */
_foo;          /* correct */
foo != Foo;
```

3.2 Keywords

The following identifiers are reserved for use as keywords, and may not be used otherwise:

3.2.1 Keyword Identifiers

```
int                                    if
float                                  else
String                                 while
matrix                                 for
bool                                   void
def                                    main
return                                 col
true                                   row
false                                  printf
printbig                               printStr
```

3.2.2 Keyword Characters

```
=      /* value binding */             >      /* greater than operator */
!      /* NOT operator  */             <      /* less than operator    */
&&     /* AND operator  */             +      /* addition operator     */
||     /* OR operator */               !=     /* not equal to operator */
<=     /* less than or equal */        >=     /* greater than or equal
(    )                                 -      /* subtraction operator */
{    }                                 ;      /* statement separator */
[    ]                                 *      /* multiplication operator */
,      /* COMMA */                     /      /* division operator */
.*     /* elementwise mult. oper. */   ./     /* elementwise div. oper. */
```

3.3 Constants

There are several kinds of constants supported by Red Pandas, as follows:

3.3.1 Integer constants

An integer constant is a sequence of one or more decimal digits, optionally preceded by a negative value. The type for *int* is 32-bit.

3.3.2 Floating point constants

A floating point constant consists of an integer part, a fractional part, and an exponent part. The integer part is a sequence of one or more digits, optionally preceded by a minus sign (-). The fractional part starts with a decimal point and continues with one or more digits. The exponent part starts with an optional plus or minus sign (+ || -) and is followed by one or more digits. This represents a power of 10 in scientific notation. Either the exponent part or fractional part can be excluded but not both. The type for *float* is 64-bit.

3.3.3 String constants

A string constant is an immutable sequence of one or more characters

3.3.4 Boolean constants

Booleans have two predefined constants for each truth value. *true* or *false*

```
true   false
```

3.4 Punctuation

Using the previously defined identifiers and constants, Red Pandas can create expressions.

3.4.1 Parentheses

The programmer can use parentheses to indicate a list of arguments in a function declaration, a function call, or a function variable declaration.

3.4.2 Brackets and Matrices

Curly brackets are used to define blocks of code such as those found in the body of a function or after an if/else statement.

```
def int foo(arg1, arg2){
   /* block */
}

foo(arg1,arg2);  /* function call */
```

Square brackets are used to access and initialize matrices. In this matrix, commas are used to separate the different elements of each row and column. Attempting to access elements that are outside of the declared range is not permitted.

Matrices can be used with the previously stated operators, such as addition and multiplication. However, when doing arithmetic on matrices, the expressions can contain only one operator at a time.

```
matrix int [2][2] foo;           /* creates a 2x2 matrix foo variable */

foo =  [ [0,1],                  /* assigns a matrix to the variable
         [1,0] ];

foo = [ [0],[1]]                 /* will throw an error, dimensions don't
                                    match /*

foo[0][1];                       /* accesses the 2nd element in the
                                    1st row and returns value of 1 */

foo[4.0][5];                      /* Non integer value in index. Error */
```

3.4.3 Semicolon and White Space

The semicolon is used to separate statements. White space is defined by the space, tab, newline, vertical tab, and form-feed characters. The use of semicolons and other separators (brackets, commas, parentheses) means that white space is ignored in Red Pandas, except to separate tokens.

```
matrix int [2][2] foo  = [ [0,1],
                           [1,0] ];
matrix int [2][2] foo2 = [ [0,1],
                           [1,0] ];
foo == foo2;                            /* returns true */

/* the following line is equivalent to the code above */

matrixfoo [[0,1],[1,0]];                /* throws ERROR matrixfoo type NOT
                                           defined, need white space here */
```

3.5 Syntax

3.5.1 Variable Declarations

```
matrix-opt var-type new-id-list semicolon
```

Var-type here is one of the primitive types or a declared function type. Matrix-opt is the optional declaration for a matrix. New-id-list is a list of one or more new (or new to scope) identifier tokens, separated by commas.

3.5.2 Function Declarations

Functions are defined this way:

```
def matrix-opt return-type new-id (matrix-opt var-type
list-opt) block
```

3.5.3 Main Function

Each program written in Red Pandas must contain the main() function. It is defined like any other function in Red Pandas, using the *def* keyword. The return type is similarly left up to the

user to specify. The main() function is meant to serve as the designed start of the program, similar to C:

```
def int main() {
...
    /* body */
...
}
```

3.6 Statements

3.6.1 Assignments

```
var-obj = actual-list semicolon
```

In an assignment statement, var-obj is defined as the rvalues from actual-list. A var-obj must be either an ID token, or a validly defined matrix location pointed to by an ID token.

The rvalues - either literals or function returns - must agree with the var-obj's declared type.

3.6.2 Blocks and Control Flow

Blocks

- A block is a curly braces-bounded set of zero or more statements:

```
def void foo() {
...
    /* a block */
...
}
```

If / else

- An If statement consists of a conditional expression followed by statements in a block. If the expression evaluates to true, the statements are evaluated.
- If an Else statement is present and the If statement evaluates to false, statements following the Else statement will be evaluated.

```
if (...)
{
/* statements */
}

else { /* if false, evaluates these statements */ }
```

While

- Similar to an If statement, While statement is a conditional expression followed by a series of one or more statements. These statements are evaluated in total and in repeating sequence for as long as the expression evaluates to true.

```
int foo;
foo = 3;

while(foo>0){foo = foo - 1;}        /* foo = 0 */
```

For

- Similarly to C / Java / etc, a For loop in Red Pandas has an expression that includes a conditional and a series of statements that are evaluated as long as the condition is evaluated as true. Unlike a While, a For loop's calling expression includes its own user defined iteration function.

```
int j; int i; matrix int [3][3] a;

a = [ [1,2,3],[3,5,6],[6,5,4] ];

for(j = 0; j < a.row; j = j + 1 ){
            for(i = 0; i < a.col; i = i + 1){
                  print(a[j][i]);
            }
}
```

3.7 Scoping

Red Pandas uses lexical scoping to define the scope of a variable. Depending on the block within which the variable is declared, no other block has access to a variable that isn't within the scope

of the block. References from lower scope level to variables that are initialized in the higher scope level are allowed, however. If there are variables with the same identifiers, then it overrides the variable that is in a higher scope.

3.7.1 Scope and Brackets

Curly brackets are used to define the blocks found in the body of a function, as well as nested blocks within the body of a function. Nested blocks include blocks defined by if/else, for, and while statements. Variables within a block will override any other variables in a higher scope that share the same identifier if the variable is assigned a new value. However, if the variable is declared explicitly within a smaller scope block, and there are variables that share the same identity outside of the block, the two variables will have different values, depending on what they are assigned, and each will be treated as a separate variable.

3.7.2 Race Conditions

Since Red Pandas overrides any variables in the higher scope that share the same identifier with a variable in a lower scope (unless explicitly declared), the programmer must be aware of potential race conditions they might create, and ensure that the statements and variable names executed are done so in a consistent sequence.

```
int i;
i = 5;   /* i = 5 */
int k;
k = 2;   /* k = 2 */

def void foo(){
    int i;
    i = 3;   /* new variable, i = 3 */
    int k;
    k = 20;  /* new variable, k = 20 */

    while (i<8){
        i++;      /* this will loop until i = 7 when foo() is executed */
    }
}

/* i = 5 , global variable * /

def int boo(){
    k = 10;        /* overrides the global variable  */
    foo();
```

```
    int j;
    j = k + i;     /* new variable, scope is function's body */
    return j;      /* returns 15 */
}

int var = boo();

/* i = 5, k = 10, new variable var = 15 */


def void goo(){
    var = j + i;   /* compiler will reject, j is not defined */
}
```

3.8 Type Conversions

In Red Pandas, the programmer must explicitly convert operands to match the data types of an expression. Red Pandas is strongly typed, and as a result, will not do any implicit type conversions. Thus, when an operator has operands of different types, the compiler will reject the expression. It is up to the programmer to convert such operands to the correct type, using the built-in functions.

```
int num1; double num2; int f;

num1 = 3;
num2 = 3.0;
f = num1 + num2  /* data types are not the same, compiler will reject */
```

3.9 Built-in Functions

3.9.1 Matrix Functions

Red Pandas has a couple of built-in functions for the matrix type. Each function is called using the identifier of the matrix being passed in, followed by the dot character, followed by the function name:

- **identifier.T**
  Transpose function. Returns the flipped version of a matrix by switching the row and columns indices.
- **.row**
  Row function. Returns the size of the rows of a matrix.
- **.col**
  Column function. Returns the size of the columns of a matrix.

3.9.2 Print Functions

Red Pandas performs printing using a variety of print functions:

- **print(int x)**
  Prints the value of an integer variable.
- **printf(float x)**
  Prints the value of a floating point number.
- **printStr(string x)**
  Prints the inputted string variable.

There is one more print function available for the user, **printbig(int x)**. It is a remnant left over from MicroC that was decided to be not taken out in case the user desired such functionality.

4. Project Plan

4.1 Planning & Development Process

Once we had all agreed on our conception of a C style language with some Pythonic flavoring and matrices, we settled on a rough schedule for development guided by the class's deliverables schedule. As our team was very unfamiliar with compiler architecture, OCaml, and language design, we tried to keep our aims modest.

In the early stages, we developed design and specifications jointly as we put together our white paper and LRM on Google Drive. Once we had finished the parser and begun development for the Hello World milestone, our roles became more formalized.

We scheduled weekly all-hands zoom calls to align on design and development planning, assigned responsibilities, and then would work on our own or in ad-hoc pairs throughout the week. The project manager maintained a Github Project Kanban board for the project to track the features next in line for development, outstanding issues, in-progress and assigned work, and completed / committed tasks.

Our development process was guided by the deliverable schedule laid out by the class - after we had put together the initial parser, we developed a skeleton for our language using the template given with MicroC. After the Hello World milestone, we began developing matrices from the ground up, working through each stage of compilation. This proved harder than expected and took joint effort from the whole team to bring the feature all the way through to code generation. Once that was completed, we worked quickly through operations for the datatype.

4.2 Programming Style Guide

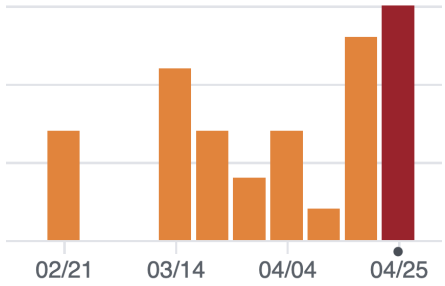
Indentation Styles — We maintained the same indentation styles for Ocaml that we found in the MicroC

Naming Conventions — Continuing the conventions of the MicroC test suite, we continued the convention of [test / fail]-[matrix(optional)]-[component] for all test files.

Git Behavior — All team members pulled at the start of each working session, pushed changes as they were completed, and alerted the team via iMessage for each major commit. If multiple people were working on a file at the same time, usually it was done so via VS Code Live Share feature, with one of the members committing the changes.

4.3 Project Timeline

January 20th — Group formation
January 26th — Group brainstorming, final project ideas
January 29th — Solidified the project idea, project proposal draft
January 31th — Project proposal finalized
February 7th — Check-in, individual proposals for syntax, presented potential source code files
February 13th — LRM drafting session, discussed TA feedback
February 20th — finalized the LRM, began work on the parser
February 23rd — Parser work session
March 1st — Weekly check-in, Spring Break
March 7th — Weekly check-in, made decisions on development environments and testing
March 16th — Group work session, began work on Hello-World
March 24th — Group work session, completed Hello-World
March 25th — Began working on extra functionality (string type, function declaration, etc.)
April 3rd — Weekly check-in: matrix types (work on the parser and the sast)
April 6th — Group work session: matrix types (work on the parser and the sast)
April 11th — Weekly check-in: matrix types, extra operator functionality, codegen, demos
April 18th — Weekly check-in: demos, presentations
April 26th — Final report writing session



GitHub Commit Activity

4.4 Roles & Responsibilities

- Amina - Tester: test suite maintenance
- Ivan - System Architect: general architecture, matrix lead (code gen)
- Myric - Project Manager: scheduling, document prep, error checking
- Rafail - Language Guru: matrix lead (scanner / parser), graphic design

4.5 Development Environment

- Environment:
    - Columbiasedwards/plt — a copy of the class's Docker image was used for compilation and run-time environment.
    - VS Code & Vim — development and collaboration tools
    - Github — main, development ("hello"), and testing branches.
- Languages
    - Ocaml — primary development language
    - Bash — shell scripting to ease testing and development
    - LLVM 10.0 — target language
- Testing
    - MicroC test suite — used as a base for developing our own unit tests
    - Menhir + other Ocaml debugging tools — parser testing & state machine visuals
    - Bash + personal testing files — for solo incremental development
    - VS Code Live Share + Zoom — for all-hands problem solving

4.6 Project Log

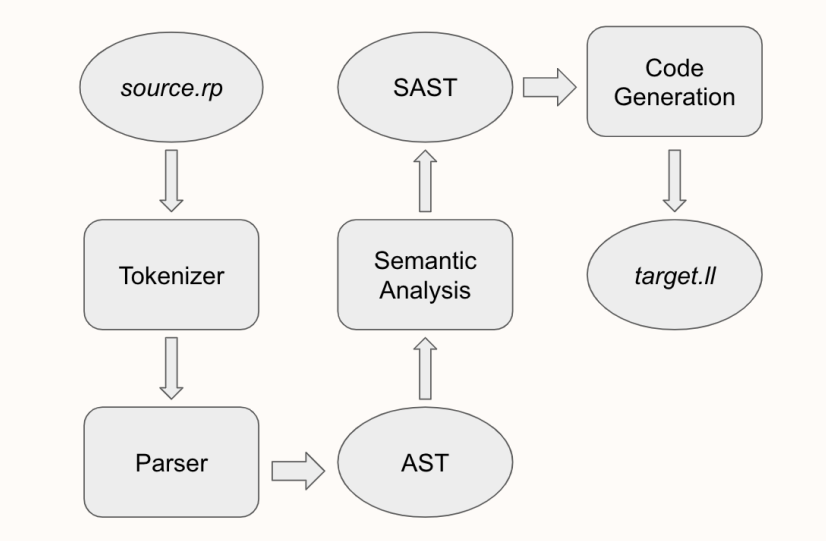
Please see appendix section 8.2 for our project log.

5. Architecture

5.1 Overview

The architecture of Red Pandas consists of four major components: the Scanner (Tokenizer), the Parser, the Semantic Checker and the Code generation. The front end of the compiler consists of the scanner and the parser, which take in the source code as input, convert it into tokens, and then they generate an abstract syntax tree (AST). The backend consists of the semantic checker and the code generation. It traverses the AST and semantically checks it, then the code is generated with the Code generator.

The entry point of the compiler is redpandas.ml, which calls all of the above components sequentially as shown in the diagram below.



5.1: Architectural Diagram

5.2 Tokenizer/Scanner *(Amina Assal, Ivan Barral, Rafail Khalilov, Myric Lehner)*

The scanner takes in the Red Pandas source code and tokenizes it into keywords, identifiers, and literals. The scanner also removes white space from the input and allows the user to make single-line or multi-line comments. If the source code includes any characters which cannot be lexed by the scanner, the scanner will throw an error.

5.3 Parser *(Amina Assal, Ivan Barral, Rafail Khalilov, Myric Lehner)*

The parser primarily consists of two files, parser.mly and ast.ml. The parser takes in the tokens generated by the scanner and using the grammar declared in parser.mly and the data types

described in ast.ml, it creates an abstract syntax tree (AST). Our parser's implementation closely follows the structure laid out in the MicroC compiler.

5.4 Semantic Checking *(Ivan Barral, Rafail Khalilov, Myric Lehner)*

The semantic checker recursively traverses the AST and creates a semantically checked syntax tree (SAST). In this step is where we do most of our type checking. This includes making sure that the types being used are valid for the operation being performed, or if values are being assigned to a variable with the correct type. For matrices, this is where we ensure that matrices consist of a single type, that each row is the same length, and assign column and row attributes of a matrix.

5.5 Code Generation *(Ivan Barral)*

If the input passes the semantic checking, the code generator takes in the SAST and returns an LLVM module. Rather than linking a C file for our matrix logic, we decided to perform these operations in the code generation step. This includes allocating the necessary space for binary matrix operations as well as performing the necessary pointer logic to access values of the matrix.

6. Test Plan

6.1 Source to Target

Source (demo.rp):

```
def void main() {

   int s;
   int i;
   int j;
   matrix int [4][4] m;

   s = 5;

   m = [[1,2,3,4],
        [5,6,7,8],
        [9,10,11,12],
        [13,14,15,16]];

   printStr("");
   printStr("s = ");
   print(s);
   printStr("");
   printStr("");

   printStr("s squared = ");
   print(s * s);
   printStr("");
   printStr("");
   printStr("m is a 4x4. m squared: ");

   m = m * m;

   for(i = 0; i < m.row; i = i + 1) {
       for(j = 0; j < m.col; j = j + 1) {
           print(m[i][j]);
       }
```

```
        printStr("");
    }
}
```

Target (demo.ll):

```
; ModuleID = 'redpandas'
source_filename = "redpandas"

@fmt = private unnamed_addr constant [4 x i8] c"%d\09\00"
@fmt.1 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt.2 = private unnamed_addr constant [4 x i8] c"%g\09\00"
@tmp = private unnamed_addr constant [24 x i8] c"Determinants are
equal:\00"
@tmp.3 = private unnamed_addr constant [1 x i8] zeroinitializer
@tmp.4 = private unnamed_addr constant [10 x i8] c"Identity:\00"
@tmp.5 = private unnamed_addr constant [1 x i8] zeroinitializer
@fmt.6 = private unnamed_addr constant [4 x i8] c"%d\09\00"
@fmt.7 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt.8 = private unnamed_addr constant [4 x i8] c"%g\09\00"

declare i32 @printf(i8*, ...)

declare i32 @printbig(i32)

define void @main() {
entry:
 %det1 = alloca i32
 %det2 = alloca i32
 %j = alloca i32
 %i = alloca i32
 %m1 = alloca [2 x [2 x i32]]
 %m2 = alloca [2 x [2 x i32]]
 %m3 = alloca [2 x [2 x i32]]
 %m4 = alloca [2 x [2 x i32]]
 %identity = alloca [2 x [2 x i32]]
 store [2 x [2 x i32]] [[2 x i32] [i32 1, i32 2], [2 x i32] [i32 3, i32
```

```
4]], [2 x [2 x i32]]* %m1
 %tmpmat = alloca [2 x [2 x i32]]
 %m11 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m1, i32 0, i32
0, i32 0
 %m12 = load i32, i32* %m11
 %tmpmat3 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %tmpmat, i32
0, i32 0, i32 0
 store i32 %m12, i32* %tmpmat3
 %m14 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m1, i32 0, i32
0, i32 1
 %m15 = load i32, i32* %m14
 %tmpmat6 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %tmpmat, i32
0, i32 1, i32 0
 store i32 %m15, i32* %tmpmat6
 %m17 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m1, i32 0, i32
1, i32 0
 %m18 = load i32, i32* %m17
 %tmpmat9 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %tmpmat, i32
0, i32 0, i32 1
 store i32 %m18, i32* %tmpmat9
 %m110 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m1, i32 0, i32
1, i32 1
 %m111 = load i32, i32* %m110
 %tmpmat12 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %tmpmat,
i32 0, i32 1, i32 1
 store i32 %m111, i32* %tmpmat12
 %tmpmat13 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %tmpmat,
i32 0
 %tmpmat14 = load [2 x [2 x i32]], [2 x [2 x i32]]* %tmpmat13
 store [2 x [2 x i32]] %tmpmat14, [2 x [2 x i32]]* %m2
 store [2 x [2 x i32]] [[2 x i32] [i32 2, i32 -1], [2 x i32] [i32 -1,
i32 1]], [2 x [2 x i32]]* %m3
 store [2 x [2 x i32]] [[2 x i32] [i32 1, i32 1], [2 x i32] [i32 1, i32
2]], [2 x [2 x i32]]* %m4
 %m315 = load [2 x [2 x i32]], [2 x [2 x i32]]* %m3
 %m416 = load [2 x [2 x i32]], [2 x [2 x i32]]* %m4
 %tmpmat17 = alloca [2 x [2 x i32]]
 %tmpval = alloca i32
 store i32 0, i32* %tmpval
```

```
 store i32 0, i32* %tmpval
 %m318 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m3, i32 0, i32
0, i32 0
 %m319 = load i32, i32* %m318
 %m420 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m4, i32 0, i32
0, i32 0
 %m421 = load i32, i32* %m420
 %tmp = mul i32 %m319, %m421
 %addtmp = load i32, i32* %tmpval
 %tmp22 = add i32 %tmp, %addtmp
 store i32 %tmp22, i32* %tmpval
 %m323 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m3, i32 0, i32
0, i32 1
 %m324 = load i32, i32* %m323
 %m425 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m4, i32 0, i32
1, i32 0
 %m426 = load i32, i32* %m425
 %tmp27 = mul i32 %m324, %m426
 %addtmp28 = load i32, i32* %tmpval
 %tmp29 = add i32 %tmp27, %addtmp28
 store i32 %tmp29, i32* %tmpval
 %tmpmat30 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %tmpmat17,
i32 0, i32 0, i32 0
 %restmp = load i32, i32* %tmpval
 store i32 %restmp, i32* %tmpmat30
 store i32 0, i32* %tmpval
 %m331 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m3, i32 0, i32
0, i32 0
 %m332 = load i32, i32* %m331
 %m433 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m4, i32 0, i32
0, i32 1
 %m434 = load i32, i32* %m433
 %tmp35 = mul i32 %m332, %m434
 %addtmp36 = load i32, i32* %tmpval
 %tmp37 = add i32 %tmp35, %addtmp36
 store i32 %tmp37, i32* %tmpval
 %m338 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m3, i32 0, i32
0, i32 1
 %m339 = load i32, i32* %m338
```

```
 %m440 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m4, i32 0, i32
1, i32 1
 %m441 = load i32, i32* %m440
 %tmp42 = mul i32 %m339, %m441
 %addtmp43 = load i32, i32* %tmpval
 %tmp44 = add i32 %tmp42, %addtmp43
 store i32 %tmp44, i32* %tmpval
 %tmpmat45 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %tmpmat17,
i32 0, i32 0, i32 1
 %restmp46 = load i32, i32* %tmpval
 store i32 %restmp46, i32* %tmpmat45
 store i32 0, i32* %tmpval
 %m347 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m3, i32 0, i32
1, i32 0
 %m348 = load i32, i32* %m347
 %m449 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m4, i32 0, i32
0, i32 0
 %m450 = load i32, i32* %m449
 %tmp51 = mul i32 %m348, %m450
 %addtmp52 = load i32, i32* %tmpval
 %tmp53 = add i32 %tmp51, %addtmp52
 store i32 %tmp53, i32* %tmpval
 %m354 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m3, i32 0, i32
1, i32 1
 %m355 = load i32, i32* %m354
 %m456 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m4, i32 0, i32
1, i32 0
 %m457 = load i32, i32* %m456
 %tmp58 = mul i32 %m355, %m457
 %addtmp59 = load i32, i32* %tmpval
 %tmp60 = add i32 %tmp58, %addtmp59
 store i32 %tmp60, i32* %tmpval
 %tmpmat61 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %tmpmat17,
i32 0, i32 1, i32 0
 %restmp62 = load i32, i32* %tmpval
 store i32 %restmp62, i32* %tmpmat61
 store i32 0, i32* %tmpval
 %m363 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m3, i32 0, i32
1, i32 0
```

```
 %m364 = load i32, i32* %m363
 %m465 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m4, i32 0, i32
0, i32 1
 %m466 = load i32, i32* %m465
 %tmp67 = mul i32 %m364, %m466
 %addtmp68 = load i32, i32* %tmpval
 %tmp69 = add i32 %tmp67, %addtmp68
 store i32 %tmp69, i32* %tmpval
 %m370 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m3, i32 0, i32
1, i32 1
 %m371 = load i32, i32* %m370
 %m472 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m4, i32 0, i32
1, i32 1
 %m473 = load i32, i32* %m472
 %tmp74 = mul i32 %m371, %m473
 %addtmp75 = load i32, i32* %tmpval
 %tmp76 = add i32 %tmp74, %addtmp75
 store i32 %tmp76, i32* %tmpval
 %tmpmat77 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %tmpmat17,
i32 0, i32 1, i32 1
 %restmp78 = load i32, i32* %tmpval
 store i32 %restmp78, i32* %tmpmat77
 %tmpmat79 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %tmpmat17,
i32 0
 %tmpmat80 = load [2 x [2 x i32]], [2 x [2 x i32]]* %tmpmat79
 store [2 x [2 x i32]] %tmpmat80, [2 x [2 x i32]]* %identity
 %m181 = load [2 x [2 x i32]], [2 x [2 x i32]]* %m1
 %calculateDet_result = call i32 @calculateDet([2 x [2 x i32]] %m181)
 store i32 %calculateDet_result, i32* %det1
 %m282 = load [2 x [2 x i32]], [2 x [2 x i32]]* %m2
 %calculateDet_result83 = call i32 @calculateDet([2 x [2 x i32]] %m282)
 store i32 %calculateDet_result83, i32* %det2
 %det184 = load i32, i32* %det1
 %det285 = load i32, i32* %det2
 %tmp86 = icmp eq i32 %det184, %det285
 br i1 %tmp86, label %then, label %else

merge:                                            ; preds = %else, %then
 %printf89 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
```

```
x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([1 x
i8], [1 x i8]* @tmp.3, i32 0, i32 0))
 %printf90 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([10
x i8], [10 x i8]* @tmp.4, i32 0, i32 0))
 store i32 0, i32* %j
 br label %while

then:                                             ; preds = %entry
 %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x
i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([24 x
i8], [24 x i8]* @tmp, i32 0, i32 0))
 %det187 = load i32, i32* %det1
 %printf88 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
x i8], [4 x i8]* @fmt, i32 0, i32 0), i32 %det187)
 br label %merge

else:                                             ; preds = %entry
 br label %merge

while:                                            ; preds = %merge102,
%merge
 %j106 = load i32, i32* %j
 %tmp107 = icmp slt i32 %j106, 2
 br i1 %tmp107, label %while_body, label %merge108

while_body:                                       ; preds = %while
 store i32 0, i32* %i
 br label %while91

while91:                                          ; preds =
%while_body92, %while_body
 %i100 = load i32, i32* %i
 %tmp101 = icmp slt i32 %i100, 2
 br i1 %tmp101, label %while_body92, label %merge102

while_body92:                                     ; preds = %while91
 %j93 = load i32, i32* %j
 %i94 = load i32, i32* %i
```

```
 %identity95 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]*
%identity, i32 0, i32 %j93, i32 %i94
 %identity96 = load i32, i32* %identity95
 %printf97 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
x i8], [4 x i8]* @fmt, i32 0, i32 0), i32 %identity96)
 %i98 = load i32, i32* %i
 %tmp99 = add i32 %i98, 1
 store i32 %tmp99, i32* %i
 br label %while91

merge102:                                         ; preds = %while91
 %printf103 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([1 x
i8], [1 x i8]* @tmp.5, i32 0, i32 0))
 %j104 = load i32, i32* %j
 %tmp105 = add i32 %j104, 1
 store i32 %tmp105, i32* %j
 br label %while

merge108:                                         ; preds = %while
 ret void
}

define i32 @calculateDet([2 x [2 x i32]] %m1) {
entry:
 %m11 = alloca [2 x [2 x i32]]
 store [2 x [2 x i32]] %m1, [2 x [2 x i32]]* %m11
 %det = alloca i32
 %m12 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m11, i32 0, i32
0, i32 0
 %m13 = load i32, i32* %m12
 %m14 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m11, i32 0, i32
1, i32 1
 %m15 = load i32, i32* %m14
 %tmp = mul i32 %m13, %m15
 %m16 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m11, i32 0, i32
0, i32 1
 %m17 = load i32, i32* %m16
 %m18 = getelementptr [2 x [2 x i32]], [2 x [2 x i32]]* %m11, i32 0, i32
```

```
1, i32 0
 %m19 = load i32, i32* %m18
 %tmp10 = mul i32 %m17, %m19
 %tmp11 = sub i32 %tmp, %tmp10
 store i32 %tmp11, i32* %det
 %det12 = load i32, i32* %det
 ret i32 %det12
}
```

6.2 Automation

In order to efficiently run our test suite, which contained around 100 tests by the end, we used the test script `testall.sh` that was provided for the microc tests and modified it to work for Red Pandas in order to run all the tests at once. We also wrote a script `rpc` that allowed us to run individual tests and programs.

6.3 Test Suites

Because the basics of Red Pandas were based on microc, we initially used tests from the microc test suite that was provided in early development to test each new feature of the compiler as it was implemented. After getting the “hello world” working for Red Pandas, we expanded the test suite to test specific features and functions of matrices.

6.3.1 Primitive data types and arithmetic functions

We tested variable declarations, assignments, and basic arithmetic operations for the primitive data types. These tests checked that the types of the variables declared matched the types of the values assigned, and that the operations used with the data types were valid for the given data types.

6.3.2 Control flow

We tested if statements and while loops to check that the conditional statements executed correctly and that the program iterated through the loop the right number of times.

6.3.3 Matrix data type and arithmetic functions

We tested variable declarations, assignments, and basic arithmetic operations for the matrix data type. Since the type of the matrix data type is strict, these tests checked that the types of the matrices declared matched the types of the values assigned to the matrices, and that the operations used with matrices only involved matrices of the same type and of valid sizes.

6.3.4 Matrix functions

We tested the two built-in matrix functions the red pandas implemented, access and transpose. For the access function, the tests checked that a valid index was given and it returned the correct value. For the transpose function, the tests checked that the size of a matrix and its transpose were valid and that the transpose was correct.

```
/usr/bin/lli

###### Testing test-add1
./redpandas.native tests/test-add1.rp > test-add1.ll
llc -relocation-model=pic test-add1.ll > test-add1.s
cc -o test-add1.exe test-add1.s printbig.o
./test-add1.exe
diff -b test-add1.out tests/test-add1.out > test-add1.diff
###### SUCCESS

###### Testing test-arith1
./redpandas.native tests/test-arith1.rp > test-arith1.ll
llc -relocation-model=pic test-arith1.ll > test-arith1.s
cc -o test-arith1.exe test-arith1.s printbig.o
./test-arith1.exe
diff -b test-arith1.out tests/test-arith1.out > test-arith1.diff
###### SUCCESS

###### Testing test-arith2
./redpandas.native tests/test-arith2.rp > test-arith2.ll
llc -relocation-model=pic test-arith2.ll > test-arith2.s
cc -o test-arith2.exe test-arith2.s printbig.o
./test-arith2.exe
diff -b test-arith2.out tests/test-arith2.out > test-arith2.diff
###### SUCCESS
```

```
###### Testing test-arith3
./redpandas.native tests/test-arith3.rp > test-arith3.ll
llc -relocation-model=pic test-arith3.ll > test-arith3.s
cc -o test-arith3.exe test-arith3.s printbig.o
./test-arith3.exe
diff -b test-arith3.out tests/test-arith3.out > test-arith3.diff
###### SUCCESS

###### Testing test-fib
./redpandas.native tests/test-fib.rp > test-fib.ll
llc -relocation-model=pic test-fib.ll > test-fib.s
cc -o test-fib.exe test-fib.s printbig.o
./test-fib.exe
diff -b test-fib.out tests/test-fib.out > test-fib.diff
###### SUCCESS

###### Testing test-float1
./redpandas.native tests/test-float1.rp > test-float1.ll
llc -relocation-model=pic test-float1.ll > test-float1.s
cc -o test-float1.exe test-float1.s printbig.o
./test-float1.exe
diff -b test-float1.out tests/test-float1.out > test-float1.diff
###### SUCCESS

###### Testing test-float2
./redpandas.native tests/test-float2.rp > test-float2.ll
llc -relocation-model=pic test-float2.ll > test-float2.s
cc -o test-float2.exe test-float2.s printbig.o
./test-float2.exe
diff -b test-float2.out tests/test-float2.out > test-float2.diff
###### SUCCESS

###### Testing test-float3
./redpandas.native tests/test-float3.rp > test-float3.ll
llc -relocation-model=pic test-float3.ll > test-float3.s
cc -o test-float3.exe test-float3.s printbig.o
./test-float3.exe
diff -b test-float3.out tests/test-float3.out > test-float3.diff
```

```
###### SUCCESS

###### Testing test-for1
./redpandas.native tests/test-for1.rp > test-for1.ll
llc -relocation-model=pic test-for1.ll > test-for1.s
cc -o test-for1.exe test-for1.s printbig.o
./test-for1.exe
diff -b test-for1.out tests/test-for1.out > test-for1.diff
###### SUCCESS

###### Testing test-for2
./redpandas.native tests/test-for2.rp > test-for2.ll
llc -relocation-model=pic test-for2.ll > test-for2.s
cc -o test-for2.exe test-for2.s printbig.o
./test-for2.exe
diff -b test-for2.out tests/test-for2.out > test-for2.diff
###### SUCCESS

###### Testing test-func1
./redpandas.native tests/test-func1.rp > test-func1.ll
llc -relocation-model=pic test-func1.ll > test-func1.s
cc -o test-func1.exe test-func1.s printbig.o
./test-func1.exe
diff -b test-func1.out tests/test-func1.out > test-func1.diff
###### SUCCESS

###### Testing test-func2
./redpandas.native tests/test-func2.rp > test-func2.ll
llc -relocation-model=pic test-func2.ll > test-func2.s
cc -o test-func2.exe test-func2.s printbig.o
./test-func2.exe
diff -b test-func2.out tests/test-func2.out > test-func2.diff
###### SUCCESS

###### Testing test-func3
./redpandas.native tests/test-func3.rp > test-func3.ll
llc -relocation-model=pic test-func3.ll > test-func3.s
cc -o test-func3.exe test-func3.s printbig.o
./test-func3.exe
```

```
diff -b test-func3.out tests/test-func3.out > test-func3.diff
###### SUCCESS

###### Testing test-func4
./redpandas.native tests/test-func4.rp > test-func4.ll
llc -relocation-model=pic test-func4.ll > test-func4.s
cc -o test-func4.exe test-func4.s printbig.o
./test-func4.exe
diff -b test-func4.out tests/test-func4.out > test-func4.diff
###### SUCCESS

###### Testing test-func5
./redpandas.native tests/test-func5.rp > test-func5.ll
llc -relocation-model=pic test-func5.ll > test-func5.s
cc -o test-func5.exe test-func5.s printbig.o
./test-func5.exe
diff -b test-func5.out tests/test-func5.out > test-func5.diff
###### SUCCESS

###### Testing test-func6
./redpandas.native tests/test-func6.rp > test-func6.ll
llc -relocation-model=pic test-func6.ll > test-func6.s
cc -o test-func6.exe test-func6.s printbig.o
./test-func6.exe
diff -b test-func6.out tests/test-func6.out > test-func6.diff
###### SUCCESS

###### Testing test-func7
./redpandas.native tests/test-func7.rp > test-func7.ll
llc -relocation-model=pic test-func7.ll > test-func7.s
cc -o test-func7.exe test-func7.s printbig.o
./test-func7.exe
diff -b test-func7.out tests/test-func7.out > test-func7.diff
###### SUCCESS

###### Testing test-func8
./redpandas.native tests/test-func8.rp > test-func8.ll
llc -relocation-model=pic test-func8.ll > test-func8.s
cc -o test-func8.exe test-func8.s printbig.o
```

```
./test-func8.exe
diff -b test-func8.out tests/test-func8.out > test-func8.diff
###### SUCCESS

###### Testing test-func9
./redpandas.native tests/test-func9.rp > test-func9.ll
llc -relocation-model=pic test-func9.ll > test-func9.s
cc -o test-func9.exe test-func9.s printbig.o
./test-func9.exe
diff -b test-func9.out tests/test-func9.out > test-func9.diff
###### SUCCESS

###### Testing test-gcd
./redpandas.native tests/test-gcd.rp > test-gcd.ll
llc -relocation-model=pic test-gcd.ll > test-gcd.s
cc -o test-gcd.exe test-gcd.s printbig.o
./test-gcd.exe
diff -b test-gcd.out tests/test-gcd.out > test-gcd.diff
###### SUCCESS

###### Testing test-gcd2
./redpandas.native tests/test-gcd2.rp > test-gcd2.ll
llc -relocation-model=pic test-gcd2.ll > test-gcd2.s
cc -o test-gcd2.exe test-gcd2.s printbig.o
./test-gcd2.exe
diff -b test-gcd2.out tests/test-gcd2.out > test-gcd2.diff
###### SUCCESS

###### Testing test-global1
./redpandas.native tests/test-global1.rp > test-global1.ll
llc -relocation-model=pic test-global1.ll > test-global1.s
cc -o test-global1.exe test-global1.s printbig.o
./test-global1.exe
diff -b test-global1.out tests/test-global1.out > test-global1.diff
###### SUCCESS

###### Testing test-global2
./redpandas.native tests/test-global2.rp > test-global2.ll
llc -relocation-model=pic test-global2.ll > test-global2.s
```

```
cc -o test-global2.exe test-global2.s printbig.o
./test-global2.exe
diff -b test-global2.out tests/test-global2.out > test-global2.diff
###### SUCCESS

###### Testing test-global3
./redpandas.native tests/test-global3.rp > test-global3.ll
llc -relocation-model=pic test-global3.ll > test-global3.s
cc -o test-global3.exe test-global3.s printbig.o
./test-global3.exe
diff -b test-global3.out tests/test-global3.out > test-global3.diff
###### SUCCESS

###### Testing test-hello
./redpandas.native tests/test-hello.rp > test-hello.ll
llc -relocation-model=pic test-hello.ll > test-hello.s
cc -o test-hello.exe test-hello.s printbig.o
./test-hello.exe
diff -b test-hello.out tests/test-hello.out > test-hello.diff
###### SUCCESS

###### Testing test-if1
./redpandas.native tests/test-if1.rp > test-if1.ll
llc -relocation-model=pic test-if1.ll > test-if1.s
cc -o test-if1.exe test-if1.s printbig.o
./test-if1.exe
diff -b test-if1.out tests/test-if1.out > test-if1.diff
###### SUCCESS

###### Testing test-if2
./redpandas.native tests/test-if2.rp > test-if2.ll
llc -relocation-model=pic test-if2.ll > test-if2.s
cc -o test-if2.exe test-if2.s printbig.o
./test-if2.exe
diff -b test-if2.out tests/test-if2.out > test-if2.diff
###### SUCCESS

###### Testing test-if3
./redpandas.native tests/test-if3.rp > test-if3.ll
```

```
llc -relocation-model=pic test-if3.ll > test-if3.s
cc -o test-if3.exe test-if3.s printbig.o
./test-if3.exe
diff -b test-if3.out tests/test-if3.out > test-if3.diff
###### SUCCESS

###### Testing test-if4
./redpandas.native tests/test-if4.rp > test-if4.ll
llc -relocation-model=pic test-if4.ll > test-if4.s
cc -o test-if4.exe test-if4.s printbig.o
./test-if4.exe
diff -b test-if4.out tests/test-if4.out > test-if4.diff
###### SUCCESS

###### Testing test-if5
./redpandas.native tests/test-if5.rp > test-if5.ll
llc -relocation-model=pic test-if5.ll > test-if5.s
cc -o test-if5.exe test-if5.s printbig.o
./test-if5.exe
diff -b test-if5.out tests/test-if5.out > test-if5.diff
###### SUCCESS

###### Testing test-if6
./redpandas.native tests/test-if6.rp > test-if6.ll
llc -relocation-model=pic test-if6.ll > test-if6.s
cc -o test-if6.exe test-if6.s printbig.o
./test-if6.exe
diff -b test-if6.out tests/test-if6.out > test-if6.diff
###### SUCCESS

###### Testing test-local1
./redpandas.native tests/test-local1.rp > test-local1.ll
llc -relocation-model=pic test-local1.ll > test-local1.s
cc -o test-local1.exe test-local1.s printbig.o
./test-local1.exe
diff -b test-local1.out tests/test-local1.out > test-local1.diff
###### SUCCESS

###### Testing test-local2
```

```
./redpandas.native tests/test-local2.rp > test-local2.ll
llc -relocation-model=pic test-local2.ll > test-local2.s
cc -o test-local2.exe test-local2.s printbig.o
./test-local2.exe
diff -b test-local2.out tests/test-local2.out > test-local2.diff
###### SUCCESS

###### Testing test-matrix-access
./redpandas.native tests/test-matrix-access.rp >
test-matrix-access.ll
llc -relocation-model=pic test-matrix-access.ll >
test-matrix-access.s
cc -o test-matrix-access.exe test-matrix-access.s printbig.o
./test-matrix-access.exe
diff -b test-matrix-access.out tests/test-matrix-access.out >
test-matrix-access.diff
###### SUCCESS

###### Testing test-matrix-add1
./redpandas.native tests/test-matrix-add1.rp > test-matrix-add1.ll
llc -relocation-model=pic test-matrix-add1.ll > test-matrix-add1.s
cc -o test-matrix-add1.exe test-matrix-add1.s printbig.o
./test-matrix-add1.exe
diff -b test-matrix-add1.out tests/test-matrix-add1.out >
test-matrix-add1.diff
###### SUCCESS

###### Testing test-matrix-mult1
./redpandas.native tests/test-matrix-mult1.rp > test-matrix-mult1.ll
llc -relocation-model=pic test-matrix-mult1.ll > test-matrix-mult1.s
cc -o test-matrix-mult1.exe test-matrix-mult1.s printbig.o
./test-matrix-mult1.exe
diff -b test-matrix-mult1.out tests/test-matrix-mult1.out >
test-matrix-mult1.diff
###### SUCCESS

###### Testing test-matrix-mult3
./redpandas.native tests/test-matrix-mult3.rp > test-matrix-mult3.ll
llc -relocation-model=pic test-matrix-mult3.ll > test-matrix-mult3.s
```

```
cc -o test-matrix-mult3.exe test-matrix-mult3.s printbig.o
./test-matrix-mult3.exe
diff -b test-matrix-mult3.out tests/test-matrix-mult3.out >
test-matrix-mult3.diff
###### SUCCESS

###### Testing test-matrix-sub1
./redpandas.native tests/test-matrix-sub1.rp > test-matrix-sub1.ll
llc -relocation-model=pic test-matrix-sub1.ll > test-matrix-sub1.s
cc -o test-matrix-sub1.exe test-matrix-sub1.s printbig.o
./test-matrix-sub1.exe
diff -b test-matrix-sub1.out tests/test-matrix-sub1.out >
test-matrix-sub1.diff
###### SUCCESS

###### Testing test-matrix-transpose1
./redpandas.native tests/test-matrix-transpose1.rp >
test-matrix-transpose1.ll
llc -relocation-model=pic test-matrix-transpose1.ll >
test-matrix-transpose1.s
cc -o test-matrix-transpose1.exe test-matrix-transpose1.s printbig.o
./test-matrix-transpose1.exe
diff -b test-matrix-transpose1.out tests/test-matrix-transpose1.out >
test-matrix-transpose1.diff
###### SUCCESS

###### Testing test-matrix-transpose2
./redpandas.native tests/test-matrix-transpose2.rp >
test-matrix-transpose2.ll
llc -relocation-model=pic test-matrix-transpose2.ll >
test-matrix-transpose2.s
cc -o test-matrix-transpose2.exe test-matrix-transpose2.s printbig.o
./test-matrix-transpose2.exe
diff -b test-matrix-transpose2.out tests/test-matrix-transpose2.out >
test-matrix-transpose2.diff
###### SUCCESS

###### Testing test-ops1
./redpandas.native tests/test-ops1.rp > test-ops1.ll
```

```
llc -relocation-model=pic test-ops1.ll > test-ops1.s
cc -o test-ops1.exe test-ops1.s printbig.o
./test-ops1.exe
diff -b test-ops1.out tests/test-ops1.out > test-ops1.diff
###### SUCCESS

###### Testing test-ops2
./redpandas.native tests/test-ops2.rp > test-ops2.ll
llc -relocation-model=pic test-ops2.ll > test-ops2.s
cc -o test-ops2.exe test-ops2.s printbig.o
./test-ops2.exe
diff -b test-ops2.out tests/test-ops2.out > test-ops2.diff
###### SUCCESS

###### Testing test-printbig
./redpandas.native tests/test-printbig.rp > test-printbig.ll
llc -relocation-model=pic test-printbig.ll > test-printbig.s
cc -o test-printbig.exe test-printbig.s printbig.o
./test-printbig.exe
diff -b test-printbig.out tests/test-printbig.out >
test-printbig.diff
###### SUCCESS

###### Testing test-var1
./redpandas.native tests/test-var1.rp > test-var1.ll
llc -relocation-model=pic test-var1.ll > test-var1.s
cc -o test-var1.exe test-var1.s printbig.o
./test-var1.exe
diff -b test-var1.out tests/test-var1.out > test-var1.diff
###### SUCCESS

###### Testing test-var2
./redpandas.native tests/test-var2.rp > test-var2.ll
llc -relocation-model=pic test-var2.ll > test-var2.s
cc -o test-var2.exe test-var2.s printbig.o
./test-var2.exe
diff -b test-var2.out tests/test-var2.out > test-var2.diff
###### SUCCESS
```

```
###### Testing test-while1
./redpandas.native tests/test-while1.rp > test-while1.ll
llc -relocation-model=pic test-while1.ll > test-while1.s
cc -o test-while1.exe test-while1.s printbig.o
./test-while1.exe
diff -b test-while1.out tests/test-while1.out > test-while1.diff
###### SUCCESS

###### Testing test-while2
./redpandas.native tests/test-while2.rp > test-while2.ll
llc -relocation-model=pic test-while2.ll > test-while2.s
cc -o test-while2.exe test-while2.s printbig.o
./test-while2.exe
diff -b test-while2.out tests/test-while2.out > test-while2.diff
###### SUCCESS

###### Testing fail-assign1
./redpandas.native < tests/fail-assign1.rp 2> fail-assign1.err >>
testall.log
diff -b fail-assign1.err tests/fail-assign1.err > fail-assign1.diff
###### SUCCESS

###### Testing fail-assign2
./redpandas.native < tests/fail-assign2.rp 2> fail-assign2.err >>
testall.log
diff -b fail-assign2.err tests/fail-assign2.err > fail-assign2.diff
###### SUCCESS

###### Testing fail-assign3
./redpandas.native < tests/fail-assign3.rp 2> fail-assign3.err >>
testall.log
diff -b fail-assign3.err tests/fail-assign3.err > fail-assign3.diff
###### SUCCESS

###### Testing fail-dead1
./redpandas.native < tests/fail-dead1.rp 2> fail-dead1.err >>
testall.log
diff -b fail-dead1.err tests/fail-dead1.err > fail-dead1.diff
###### SUCCESS
```

```
###### Testing fail-dead2
./redpandas.native < tests/fail-dead2.rp 2> fail-dead2.err >>
testall.log
diff -b fail-dead2.err tests/fail-dead2.err > fail-dead2.diff
###### SUCCESS

###### Testing fail-expr1
./redpandas.native < tests/fail-expr1.rp 2> fail-expr1.err >>
testall.log
diff -b fail-expr1.err tests/fail-expr1.err > fail-expr1.diff
###### SUCCESS

###### Testing fail-expr2
./redpandas.native < tests/fail-expr2.rp 2> fail-expr2.err >>
testall.log
diff -b fail-expr2.err tests/fail-expr2.err > fail-expr2.diff
###### SUCCESS

###### Testing fail-expr3
./redpandas.native < tests/fail-expr3.rp 2> fail-expr3.err >>
testall.log
diff -b fail-expr3.err tests/fail-expr3.err > fail-expr3.diff
###### SUCCESS

###### Testing fail-float1
./redpandas.native < tests/fail-float1.rp 2> fail-float1.err >>
testall.log
diff -b fail-float1.err tests/fail-float1.err > fail-float1.diff
###### SUCCESS

###### Testing fail-float2
./redpandas.native < tests/fail-float2.rp 2> fail-float2.err >>
testall.log
diff -b fail-float2.err tests/fail-float2.err > fail-float2.diff
###### SUCCESS

###### Testing fail-for1
./redpandas.native < tests/fail-for1.rp 2> fail-for1.err >>
```

```
testall.log
diff -b fail-for1.err tests/fail-for1.err > fail-for1.diff
###### SUCCESS

###### Testing fail-for2
./redpandas.native < tests/fail-for2.rp 2> fail-for2.err >>
testall.log
diff -b fail-for2.err tests/fail-for2.err > fail-for2.diff
###### SUCCESS

###### Testing fail-for3
./redpandas.native < tests/fail-for3.rp 2> fail-for3.err >>
testall.log
diff -b fail-for3.err tests/fail-for3.err > fail-for3.diff
###### SUCCESS

###### Testing fail-for4
./redpandas.native < tests/fail-for4.rp 2> fail-for4.err >>
testall.log
diff -b fail-for4.err tests/fail-for4.err > fail-for4.diff
###### SUCCESS

###### Testing fail-for5
./redpandas.native < tests/fail-for5.rp 2> fail-for5.err >>
testall.log
diff -b fail-for5.err tests/fail-for5.err > fail-for5.diff
###### SUCCESS

###### Testing fail-func1
./redpandas.native < tests/fail-func1.rp 2> fail-func1.err >>
testall.log
diff -b fail-func1.err tests/fail-func1.err > fail-func1.diff
###### SUCCESS

###### Testing fail-func2
./redpandas.native < tests/fail-func2.rp 2> fail-func2.err >>
testall.log
diff -b fail-func2.err tests/fail-func2.err > fail-func2.diff
###### SUCCESS
```

```
###### Testing fail-func3
./redpandas.native < tests/fail-func3.rp 2> fail-func3.err >>
testall.log
diff -b fail-func3.err tests/fail-func3.err > fail-func3.diff
###### SUCCESS

###### Testing fail-func4
./redpandas.native < tests/fail-func4.rp 2> fail-func4.err >>
testall.log
diff -b fail-func4.err tests/fail-func4.err > fail-func4.diff
###### SUCCESS

###### Testing fail-func5
./redpandas.native < tests/fail-func5.rp 2> fail-func5.err >>
testall.log
diff -b fail-func5.err tests/fail-func5.err > fail-func5.diff
###### SUCCESS

###### Testing fail-func6
./redpandas.native < tests/fail-func6.rp 2> fail-func6.err >>
testall.log
diff -b fail-func6.err tests/fail-func6.err > fail-func6.diff
###### SUCCESS

###### Testing fail-func7
./redpandas.native < tests/fail-func7.rp 2> fail-func7.err >>
testall.log
diff -b fail-func7.err tests/fail-func7.err > fail-func7.diff
###### SUCCESS

###### Testing fail-func8
./redpandas.native < tests/fail-func8.rp 2> fail-func8.err >>
testall.log
diff -b fail-func8.err tests/fail-func8.err > fail-func8.diff
###### SUCCESS

###### Testing fail-func9
./redpandas.native < tests/fail-func9.rp 2> fail-func9.err >>
```

```
testall.log
diff -b fail-func9.err tests/fail-func9.err > fail-func9.diff
###### SUCCESS

###### Testing fail-global1
./redpandas.native < tests/fail-global1.rp 2> fail-global1.err >>
testall.log
diff -b fail-global1.err tests/fail-global1.err > fail-global1.diff
###### SUCCESS

###### Testing fail-global2
./redpandas.native < tests/fail-global2.rp 2> fail-global2.err >>
testall.log
diff -b fail-global2.err tests/fail-global2.err > fail-global2.diff
###### SUCCESS

###### Testing fail-if1
./redpandas.native < tests/fail-if1.rp 2> fail-if1.err >> testall.log
diff -b fail-if1.err tests/fail-if1.err > fail-if1.diff
###### SUCCESS

###### Testing fail-if2
./redpandas.native < tests/fail-if2.rp 2> fail-if2.err >> testall.log
diff -b fail-if2.err tests/fail-if2.err > fail-if2.diff
###### SUCCESS

###### Testing fail-if3
./redpandas.native < tests/fail-if3.rp 2> fail-if3.err >> testall.log
diff -b fail-if3.err tests/fail-if3.err > fail-if3.diff
###### SUCCESS

###### Testing fail-matrix-access
./redpandas.native < tests/fail-matrix-access.rp 2>
fail-matrix-access.err >> testall.log
diff -b fail-matrix-access.err tests/fail-matrix-access.err >
fail-matrix-access.diff
###### SUCCESS

###### Testing fail-matrix-add1
```

```
./redpandas.native < tests/fail-matrix-add1.rp 2>
fail-matrix-add1.err >> testall.log
diff -b fail-matrix-add1.err tests/fail-matrix-add1.err >
fail-matrix-add1.diff
###### SUCCESS

###### Testing fail-matrix-add2
./redpandas.native < tests/fail-matrix-add2.rp 2>
fail-matrix-add2.err >> testall.log
diff -b fail-matrix-add2.err tests/fail-matrix-add2.err >
fail-matrix-add2.diff
###### SUCCESS

###### Testing fail-matrix-add3
./redpandas.native < tests/fail-matrix-add3.rp 2>
fail-matrix-add3.err >> testall.log
diff -b fail-matrix-add3.err tests/fail-matrix-add3.err >
fail-matrix-add3.diff
###### SUCCESS

###### Testing fail-matrix-add4
./redpandas.native < tests/fail-matrix-add4.rp 2>
fail-matrix-add4.err >> testall.log
diff -b fail-matrix-add4.err tests/fail-matrix-add4.err >
fail-matrix-add4.diff
###### SUCCESS

###### Testing fail-matrix-arith
./redpandas.native < tests/fail-matrix-arith.rp 2>
fail-matrix-arith.err >> testall.log
diff -b fail-matrix-arith.err tests/fail-matrix-arith.err >
fail-matrix-arith.diff
###### SUCCESS

###### Testing fail-matrix-mult1
./redpandas.native < tests/fail-matrix-mult1.rp 2>
fail-matrix-mult1.err >> testall.log
diff -b fail-matrix-mult1.err tests/fail-matrix-mult1.err >
fail-matrix-mult1.diff
```

```
###### SUCCESS

###### Testing fail-matrix-mult2
./redpandas.native < tests/fail-matrix-mult2.rp 2>
fail-matrix-mult2.err >> testall.log
diff -b fail-matrix-mult2.err tests/fail-matrix-mult2.err >
fail-matrix-mult2.diff
###### SUCCESS

###### Testing fail-matrix-mult3
./redpandas.native < tests/fail-matrix-mult3.rp 2>
fail-matrix-mult3.err >> testall.log
diff -b fail-matrix-mult3.err tests/fail-matrix-mult3.err >
fail-matrix-mult3.diff
###### SUCCESS

###### Testing fail-matrix-mult4
./redpandas.native < tests/fail-matrix-mult4.rp 2>
fail-matrix-mult4.err >> testall.log
diff -b fail-matrix-mult4.err tests/fail-matrix-mult4.err >
fail-matrix-mult4.diff
###### SUCCESS

###### Testing fail-matrix-sub1
./redpandas.native < tests/fail-matrix-sub1.rp 2>
fail-matrix-sub1.err >> testall.log
diff -b fail-matrix-sub1.err tests/fail-matrix-sub1.err >
fail-matrix-sub1.diff
###### SUCCESS

###### Testing fail-matrix-sub2
./redpandas.native < tests/fail-matrix-sub2.rp 2>
fail-matrix-sub2.err >> testall.log
diff -b fail-matrix-sub2.err tests/fail-matrix-sub2.err >
fail-matrix-sub2.diff
###### SUCCESS

###### Testing fail-matrix-sub3
./redpandas.native < tests/fail-matrix-sub3.rp 2>
```

```
fail-matrix-sub3.err >> testall.log
diff -b fail-matrix-sub3.err tests/fail-matrix-sub3.err >
fail-matrix-sub3.diff
###### SUCCESS

###### Testing fail-matrix-sub4
./redpandas.native < tests/fail-matrix-sub4.rp 2>
fail-matrix-sub4.err >> testall.log
diff -b fail-matrix-sub4.err tests/fail-matrix-sub4.err >
fail-matrix-sub4.diff
###### SUCCESS

###### Testing fail-matrix-transpose
./redpandas.native < tests/fail-matrix-transpose.rp 2>
fail-matrix-transpose.err >> testall.log
diff -b fail-matrix-transpose.err tests/fail-matrix-transpose.err >
fail-matrix-transpose.diff
###### SUCCESS

###### Testing fail-nomain
./redpandas.native < tests/fail-nomain.rp 2> fail-nomain.err >>
testall.log
diff -b fail-nomain.err tests/fail-nomain.err > fail-nomain.diff
###### SUCCESS

###### Testing fail-print
./redpandas.native < tests/fail-print.rp 2> fail-print.err >>
testall.log
diff -b fail-print.err tests/fail-print.err > fail-print.diff
###### SUCCESS

###### Testing fail-printb
./redpandas.native < tests/fail-printb.rp 2> fail-printb.err >>
testall.log
diff -b fail-printb.err tests/fail-printb.err > fail-printb.diff
###### SUCCESS

###### Testing fail-printbig
./redpandas.native < tests/fail-printbig.rp 2> fail-printbig.err >>
```

```
testall.log
diff -b fail-printbig.err tests/fail-printbig.err >
fail-printbig.diff
###### SUCCESS

###### Testing fail-return1
./redpandas.native < tests/fail-return1.rp 2> fail-return1.err >>
testall.log
diff -b fail-return1.err tests/fail-return1.err > fail-return1.diff
###### SUCCESS

###### Testing fail-return2
./redpandas.native < tests/fail-return2.rp 2> fail-return2.err >>
testall.log
diff -b fail-return2.err tests/fail-return2.err > fail-return2.diff
###### SUCCESS

###### Testing fail-while1
./redpandas.native < tests/fail-while1.rp 2> fail-while1.err >>
testall.log
diff -b fail-while1.err tests/fail-while1.err > fail-while1.diff
###### SUCCESS

###### Testing fail-while2
./redpandas.native < tests/fail-while2.rp 2> fail-while2.err >>
testall.log
diff -b fail-while2.err tests/fail-while2.err > fail-while2.diff
###### SUCCESS
```

6.4 Testing Roles

Ivan modified the microc test suite to create the skeleton of our test suite and wrote the rpc script for running individual tests. Amina designed the test cases for all additional tests regarding matrices and matrix functions and reported any issues to the rest of the team, who worked to resolve them. Rafail designed the first test case, hello-world.

7. Lessons Learned

Amina: Aside from the many technical things I learned over the course of this project, I learned the most about workflow and project management. Having assigned roles made it clear the tasks each member of the team was responsible for, even without explicitly being assigned those tasks. The only potential issue with this kind of workflow where everyone is working on a separate part of the project is that there could be confusion about what has been completed and how. One thing I really appreciated about my group is that whenever there was a major push to our github repository, it would be summarized and sent to our group chat so that everyone was updated about the status of our project. One thing I would change about the workflow is rather than working in less frequent and more involved sprints, I think we would have benefited more from working in more frequent and less involved sprints so that there could have been opportunities to revise our project and catch any bugs.

Ivan: I learned several lessons from this project, from both a technical and team perspective. From a technical perspective, this course was my first time using Ocaml, so applying the lessons we learned from our first homework and the MicroC compiler into our own compiler. This had quite a large learning curve, especially when trying to figure out how to create matrix logic with LLVM. One major aspect of creating a language is realizing that each minor decision may have a major impact on the code. This was especially relevant when we ran into errors in our code and would accidentally create more errors when trying to patch it up. My advice on the technical side would be to try to avoid trial and error and instead try to fully understand what your code is doing to avoid creating more errors. From a team perspective, I learned how crucial communication is to creating a good product. Constantly updating the team on how the project was progressing and asking questions was vital for success. On this front, I would advise students to meet with their team as often as possible to try to generate constant improvements to the language. I definitely regretted putting off the project a bit around the midpoint of our timeline and not being able to add all the features I wanted to our language.

Myric: I learned an enormous amount from this class and this project. Working on this scale where automation was key to successful organized unit testing was fascinating and gave me an appreciation for the tools that can further help automate the automation process. Working incrementally is clearly key, but also researching the tools at our disposal thoroughly ahead of time, because without those, incrementally moving through the compiler stages is very difficult. Now that I more fully understand the scale of the project, I wish I had tackled the material much more aggressively earlier on. Since we are all working remotely, I would also have scheduled at

least another set of regular drop-in meetings, since otherwise communication was only through iMessage.

Rafail: Getting this project done incrementally is the key. Not only does that help maintain the momentum, but it also helps you focus on one feature at a time, meaning you make better design decisions. I am grateful for my team, as I feel like we all tried our best to get as much as possible done at any given point, even if there weren't any deadlines in sight. That made the project a whole lot stressful long-term. However, there is really no such thing as starting early with this project. The more you get done early, the more time you have to implement all the cool features you want your language to have. Overall, as long as you meet regularly (meeting at least once a week) and communicate your concerns clearly, the work splits itself over time rather naturally. Also, if you do the project one step at a time, it will also help you solidify the knowledge you gained, meaning you won't have to prepare as much for the final exam. In retrospect, my biggest lesson is don't be afraid to let this project be a learning process. Not everything will make sense initially, but it will eventually. For me, that meant being comfortable asking other team members questions about the compiler architecture and whatever else I struggled with understanding. Doing that is helpful for everyone! Programming in OCaml can definitely be challenging at first, so truly understanding hw1 is crucial. In terms of actually building the compiler, for me, as the language guru, the hardest part was making sure that I had a solid idea of what I would like the parser and the grammar to look like, while predicting the different possibilities that could come as a result of the specific design choices – there is nothing worse than going back to change something in the parser after realizing your mistake while working on semantic checking or code generation. However, if you plan diligently and put in enough work from the start, this project is actually quite fun and totally manageable.

8. Appendices

8.1 Project Code

8.1.1 Makefile

```
# "make test" Compiles everything and runs the regression tests

.PHONY : test
test : all testall.sh
   ./testall.sh

# "make all" builds the executable as well as the "printbig" library designed
# to test linking external code

.PHONY : all
all : redpandas.native printbig.o

# "make redpandas.native" compiles the compiler
#
# The _tags file controls the operation of ocamlbuild, e.g., by including
# packages, enabling warnings
#
# See https://github.com/ocaml/ocamlbuild/blob/master/manual/manual.adoc

redpandas.native :
   opam config exec -- \
   ocamlbuild -use-ocamlfind redpandas.native

# "make clean" removes all generated files

.PHONY : clean
clean :
   ocamlbuild -clean
   rm -rf testall.log ocamlllvm *.diff *.err *.ll *.s *.exe

.PHONY : smallclean
smallclean :
   rm -rf testall.log *.diff *.err *.ll

# Testing the "printbig" example

printbig : printbig.c
   cc -o printbig -DBUILD_TEST printbig.c

# Building the tarball
```

```
TESTS = \
 add1 arith1 arith2 arith3 fib float1 float2 float3 for1 for2 func1 \
 func2 func3 func4 func5 func6 func7 func8 func9 gcd2 gcd global1 \
 global2 global3 hello if1 if2 if3 if4 if5 if6 local1 local2 ops1 \
 ops2 printbig var1 var2 while1 while2

FAILS = \
 assign1 assign2 assign3 dead1 dead2 expr1 expr2 expr3 float1 float2 \
 for1 for2 for3 for4 for5 func1 func2 func3 func4 func5 func6 func7 \
 func8 func9 global1 global2 if1 if2 if3 nomain printbig printb print \
 return1 return2 while1 while2

TESTFILES = $(TESTS:%=test-%.rp) $(TESTS:%=test-%.out) \
       $(FAILS:%=fail-%.rp) $(FAILS:%=fail-%.err)

TARFILES = ast.ml sast.ml codegen.ml Makefile _tags redpandas.ml parser.mly \
   README.md scanner.mll semant.ml testall.sh \
   printbig.c arcade-font.pbm font2c \
   Dockerfile hello_world.rp hello_world_2.rp \
   $(TESTFILES:%=tests/%)

redpandas.tar.gz : $(TARFILES)
   cd .. && tar czf Red-Pandas/redpandas.tar.gz \
       $(TARFILES:%=Red-Pandas/%)
```

8.1.2 scanner.mll

```
(* Ocamllex scanner for Red-Pandas *)

{ open Parser }

let digit = ['0' - '9']
let digits = digit+

rule token = parse
  [' ' '\t' '\r' '\n'] { token lexbuf } (* Whitespace *)
| "/*"     { comment lexbuf }           (* Comments *)
| "//"     { line_comment lexbuf}
| '('      { LPAREN }
| '.'      { PERIOD }
| ')'      { RPAREN }
| '{'      { LBRACE }
| '}'      { RBRACE }
```

```
| '['      { LBRACK }
| ']'      { RBRACK }
| ';'      { SEMI }
| ','      { COMMA }
| '+'      { PLUS }
| '-'      { MINUS }
| '*'      { TIMES }
| '/'      { DIVIDE }
| ".*"     { ELTIMES }
| "./"     { ELDIVIDE }
| "T"      { TRANSP }
| '='      { ASSIGN }
| "=="     { EQ }
| "!="     { NEQ }
| '<'      { LT }
| "<="     { LEQ }
| ">"      { GT }
| ">="     { GEQ }
| "&&"     { AND }
| "||"     { OR }
| "!"      { NOT }
| "if"     { IF }
| "else"   { ELSE }
| "for"    { FOR }
| "while"  { WHILE }
| "return" { RETURN }
| "int"    { INT }
| "bool"   { BOOL }
| "float"  { FLOAT }
| "void"   { VOID }
| "String" { STRING }
| "matrix" { MATRIX }
| "true"   { BLIT(true)  }
| "false"  { BLIT(false) }
| "col"    { COL }
| "row"    { ROW }
| "def"    { DEF }
| digits as lxm { LITERAL(int_of_string lxm) }
| digits '.'  digit* ( ['e' 'E'] ['+' '-']? digits )? as lxm { FLIT(lxm) }
| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']*     as lxm { ID(lxm) }
| '"' (([^ '"'] | "\\\"")* as lxm) '"' { STRLIT(lxm) }
| eof { EOF }
| _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }
```

```
and comment = parse
  "*/" { token lexbuf }
| _    { comment lexbuf }

and line_comment = parse
  "\n" { token lexbuf }
```

8.1.3 parser.mly

```
%{
open Ast
let parse_error s =
      begin
        try
          let start_pos = Parsing.symbol_start_pos ()
          and end_pos = Parsing.symbol_end_pos () in
          Printf.printf "File \"%s\", line %d, characters %d-%d: \n"
            start_pos.pos_fname
            start_pos.pos_lnum
            (start_pos.pos_cnum - start_pos.pos_bol)
            (end_pos.pos_cnum - start_pos.pos_bol)
        with Invalid_argument(_) -> ()
      end;
      Printf.printf "Syntax error: %s\n" s;
      raise Parsing.Parse_error

%}

%token SEMI LPAREN RPAREN LBRACE RBRACE COMMA
%token LBRACK RBRACK PERIOD
%token PLUS MINUS TIMES DIVIDE ELTIMES ELDIVIDE ASSIGN
%token NOT EQ NEQ LT LEQ GT GEQ AND OR
%token RETURN IF ELSE FOR WHILE
%token STRING MATRIX INT BOOL FLOAT VOID
%token TRANSP
%token COL ROW
%token DEF
%token <int> LITERAL
%token <bool> BLIT
%token <string> ID FLIT
```

```
%token <string> STRLIT
%token EOF

%start program
%type <Ast.program> program

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

%%

program:
  decls EOF { $1 }

decls:
   /* nothing */ { ([], [])               }
 | decls vdecl { (($2 :: fst $1), snd $1) }
 | decls fdecl { (fst $1, ($2 :: snd $1)) }

fdecl:
   DEF typ ID LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list RBRACE
     { { typ = $2;
       fname = $3;
       formals = List.rev $5;
       locals = List.rev $8;
       body = List.rev $9 } }

formals_opt:
    /* nothing */ { [] }
  | formal_list   { $1 }

formal_list:
    typ ID                   { [($1,$2)]     }
  | formal_list COMMA typ ID { ($3,$4) :: $1 }
```

```
typ:
    INT        { Int           }
  | BOOL       { Bool          }
  | FLOAT      { Float         }
  | VOID       { Void          }
  | STRING     { String        }
  | MATRIX typ LBRACK LITERAL RBRACK LBRACK LITERAL RBRACK   { Matrix($2,
$4, $7) }

vdecl_list:
    /* nothing */    { [] }
  | vdecl_list vdecl { $2 :: $1 }

vdecl:
   typ ID SEMI { ($1, $2) }

stmt_list:
    /* nothing */  { [] }
  | stmt_list stmt { $2 :: $1 }

stmt:
    expr SEMI                               { Expr $1               }
  | RETURN expr_opt SEMI                    { Return $2             }
  | LBRACE stmt_list RBRACE                 { Block(List.rev $2)    }
  | IF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5, Block([])) }
  | IF LPAREN expr RPAREN stmt ELSE stmt    { If($3, $5, $7)        }
  | FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN stmt
                                            { For($3, $5, $7, $9)   }
  | WHILE LPAREN expr RPAREN stmt           { While($3, $5)         }

expr_opt:
    /* nothing */ { Noexpr }
  | expr          { $1 }

expr:
    LITERAL          { Literal($1)            }
  | FLIT             { Fliteral($1)           }
  | BLIT             { BoolLit($1)            }
  | ID               { Id($1)                 }
  | STRLIT           { StrLit($1)             }
  | expr PLUS   expr { Binop($1, Add,   $3)   }
  | expr MINUS  expr { Binop($1, Sub,   $3)   }
```

```
  | expr TIMES  expr { Binop($1, Mult,  $3)   }
  | expr DIVIDE expr { Binop($1, Div,   $3)   }
  | expr ELTIMES expr  { Binop($1, Elmult,   $3) }
  | expr ELDIVIDE expr { Binop($1, Eldiv,   $3)  }
  | expr EQ     expr { Binop($1, Equal, $3)   }
  | expr NEQ    expr { Binop($1, Neq,   $3)   }
  | expr LT     expr { Binop($1, Less,  $3)   }
  | expr LEQ    expr { Binop($1, Leq,   $3)   }
  | expr GT     expr { Binop($1, Greater, $3) }
  | expr GEQ    expr { Binop($1, Geq,   $3)   }
  | expr AND    expr { Binop($1, And,   $3)   }
  | expr OR     expr { Binop($1, Or,    $3)   }
  | MINUS expr %prec NOT { Unop(Neg, $2)      }
  | NOT expr         { Unop(Not, $2)          }
  | expr ASSIGN expr   { Assign($1, $3)       }
  | ID LPAREN args_opt RPAREN { Call($1, $3)  }
  | LPAREN expr RPAREN        { $2            }
  | LBRACK mat_opt RBRACK     { Mat($2)       }
  | ID PERIOD COL             { Col($1)       }
  | ID PERIOD ROW             { Row($1)       }
  | ID PERIOD TRANSP          { Tran($1)      }
  | ID LBRACK expr RBRACK LBRACK expr RBRACK
                     { Access($1, $3, $6)    }

mat_opt:
    /* nothing */ { [] }
  | row_list      { List.rev $1 }

row_list:
    LBRACK row_expr RBRACK                { [(List.rev $2)]        }
  | row_list COMMA LBRACK row_expr RBRACK {  (List.rev $4) :: $1 }

row_expr:
    /* nothing */            { []       }
  | expr                     { [$1]    }
  | row_expr COMMA expr      { $3 :: $1 }

args_opt:
    /* nothing */ { [] }
  | args_list  { List.rev $1 }

args_list:
    expr                   { [$1] }
```

```
  | args_list COMMA expr { $3 :: $1 }
```

8.1.4 ast.ml

```
(* Abstract Syntax Tree and functions for printing it *)

type op = Add | Sub | Mult | Div | Equal | Neq | Less | Leq | Greater | Geq
|
          And | Or | Elmult | Eldiv

type uop = Neg | Not

type expr =
    Literal of int
  | Fliteral of string
  | BoolLit of bool
  | StrLit of string
  | Id of string
  | Binop of expr * op * expr
  | Unop of uop * expr
  | Assign of expr * expr
  | Call of string * expr list
  | Mat of expr list list
  | Col of string
  | Row of string
  | Tran of string
  | Access of string * expr * expr
  | Noexpr

type typ =
    Void
  | Int
  | Bool
  | Float
  | String
  | Matrix of typ * int * int

type bind = typ * string

type stmt =
    Block of stmt list
```

```
  | Expr of expr
  | Return of expr
  | If of expr * stmt * stmt
  | For of expr * expr * expr * stmt
  | While of expr * stmt

type func_decl = {
    typ : typ;
    fname : string;
    formals : bind list;
    locals : bind list;
    body : stmt list;
  }

type program = bind list * func_decl list

(* Pretty-printing functions *)

let string_of_op = function
    Add -> "+"
  | Sub -> "-"
  | Mult -> "*"
  | Div -> "/"
  | Elmult -> ".*"
  | Eldiv -> "./"
  | Equal -> "=="
  | Neq -> "!="
  | Less -> "<"
  | Leq -> "<="
  | Greater -> ">"
  | Geq -> ">="
  | And -> "&&"
  | Or -> "||"

let string_of_uop = function
    Neg -> "-"
  | Not -> "!"

let rec string_of_expr = function
    Literal(l) -> string_of_int l
  | Fliteral(l) -> l
  | BoolLit(true) -> "true"
  | BoolLit(false) -> "false"
```

```
  | StrLit(l) -> "\"" ^ (String.escaped l) ^ "\""
  | Mat(_) -> "matLit"
  | Id(s) -> s
  | Binop(e1, o, e2) ->
      string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
  | Unop(o, e) -> string_of_uop o ^ string_of_expr e
  | Assign(v, e) -> string_of_expr v ^ " = " ^ string_of_expr e
  | Call(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | Col(s) -> s ^ ".col"
  | Row(s) -> s ^ ".row"
  | Tran(s) -> s ^ ".T"
  | Access(s, r, c) -> s ^ "[" ^ string_of_expr r ^ "]" ^ "[" ^
string_of_expr c ^ "]"
  | Noexpr -> ""

let rec string_of_stmt = function
    Block(stmts) ->
      "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
  | Expr(expr) -> string_of_expr expr ^ ";\n";
  | Return(expr) -> "return " ^ string_of_expr expr ^ ";\n";
  | If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^
string_of_stmt s
  | If(e, s1, s2) ->  "if (" ^ string_of_expr e ^ ")\n" ^
      string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2
  | For(e1, e2, e3, s) ->
      "for (" ^ string_of_expr e1  ^ " ; " ^ string_of_expr e2 ^ " ; " ^
      string_of_expr e3  ^ ") " ^ string_of_stmt s
  | While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^ string_of_stmt s

let string_of_typ = function
    Int -> "int"
  | Bool -> "bool"
  | Float -> "float"
  | Void -> "void"
  | String -> "String"
  | Matrix(_, r, c) -> "matrix [" ^ (string_of_int r) ^ "][" ^
string_of_int c ^ "]"

let string_of_vdecl (t, id) = string_of_typ t ^ " " ^ id ^ ";\n"

let string_of_fdecl fdecl =
  "def" ^ " " ^ string_of_typ fdecl.typ ^ " " ^
```

```
  fdecl.fname ^ "(" ^ String.concat ", " (List.map snd fdecl.formals) ^
  ")\n{\n" ^
  String.concat "" (List.map string_of_vdecl fdecl.locals) ^
  String.concat "" (List.map string_of_stmt fdecl.body) ^
  "}\n"

let string_of_program (vars, funcs) =
  String.concat "" (List.map string_of_vdecl vars) ^ "\n" ^
  String.concat "\n" (List.map string_of_fdecl funcs)
```

8.1.5 semant.ml

```
(* Semantic checking for the RedPandas compiler *)

open Ast
open Sast

module StringMap = Map.Make(String)

(* Semantic checking of the AST. Returns an SAST if successful,
   throws an exception if something is wrong.

   Check each global variable, then check each function *)

let check (globals, functions) =

  (* Verify a list of bindings has no void types or duplicate names *)
  let check_binds (kind : string) (binds : bind list) =
    List.iter (function
      (Void, b) -> raise (Failure ("illegal void " ^ kind ^ " " ^ b))
    | _ -> ()) binds;
    let rec dups = function
        [] -> ()
      |     ((_,n1) :: (_,n2) :: _) when n1 = n2 ->
        raise (Failure ("duplicate " ^ kind ^ " " ^ n1))
      | _ :: t -> dups t
    in dups (List.sort (fun (_,a) (_,b) -> compare a b) binds)
  in

  (**** Check global variables ****)

  check_binds "global" globals;

  (**** Check functions ****)
```

```
(* Collect function declarations for built-in functions: no bodies *)
let built_in_decls =
  let add_bind map (name, ty) = StringMap.add name {
    typ = Void;
    fname = name;
    formals = [(ty, "x")];
    locals = []; body = [] } map
  in List.fold_left add_bind StringMap.empty [ ("print", Int);
                                               ("printb", Bool);
                                               ("printf", Float);
                                               ("printbig", Int);
                                ("printStr", String) ]
in

(* Add function name to symbol table *)
let add_func map fd =
  let built_in_err = "function " ^ fd.fname ^ " may not be defined"
  and dup_err = "duplicate function " ^ fd.fname
  and make_err er = raise (Failure er)
  and n = fd.fname (* Name of the function *)
  in match fd with (* No duplicate functions or redefinitions of built-ins *)
       _ when StringMap.mem n built_in_decls -> make_err built_in_err
     | _ when StringMap.mem n map -> make_err dup_err
     | _ ->  StringMap.add n fd map
in

(* Collect all function names into one symbol table *)
let function_decls = List.fold_left add_func built_in_decls functions
in

(* Return a function from our symbol table *)
let find_func s =
  try StringMap.find s function_decls
  with Not_found -> raise (Failure ("unrecognized function " ^ s))
in

let _ = find_func "main" in (* Ensure "main" is defined *)

let check_function func =
  (* Make sure no formals or locals are void or duplicates *)
  check_binds "formal" func.formals;
  check_binds "local" func.locals;

  (* Raise an exception if the given rvalue type cannot be assigned to
     the given lvalue type *)
  let check_assign lvaluet rvaluet err =
     if lvaluet = rvaluet then lvaluet else raise (Failure err)
```

```
  in

  (* Build local symbol table of variables for this function *)
  let symbols = List.fold_left (fun m (ty, name) -> StringMap.add name ty m)
                    StringMap.empty (globals @ func.formals @ func.locals )
  in

  (* Return a variable from our local symbol table *)
  let type_of_identifier s =
    try StringMap.find s symbols
    with Not_found -> raise (Failure ("undeclared identifier " ^ s))
  in

  (* Return a semantically-checked expression, i.e., with a type *)
  let rec expr = function
      Literal  l -> (Int, SLiteral l)
    | Fliteral l -> (Float, SFliteral l)
    | BoolLit l  -> (Bool, SBoolLit l)
    | StrLit l   -> (String, SStrLit l)
    | Noexpr     -> (Void, SNoexpr)
    | Id s       -> (type_of_identifier s, SId s)
    | Assign(var, e) as ex ->
        let (lt, var') = expr var
        and (rt, e') = expr e in
        let err = "illegal assignment " ^ string_of_typ lt ^ " = " ^
          string_of_typ rt ^ " in " ^ string_of_expr ex
        in (check_assign lt rt err, SAssign((lt, var'), (rt, e')))
    | Unop(op, e) as ex ->
        let (t, e') = expr e in
        let ty = match op with
          Neg when t = Int || t = Float -> t
        | Not when t = Bool -> Bool
        | _ -> raise (Failure ("illegal unary operator " ^
                               string_of_uop op ^ string_of_typ t ^
                               " in " ^ string_of_expr ex))
        in (ty, SUnop(op, (t, e')))
    | Binop(e1, op, e2) as e ->
        let (t1, e1') = expr e1
        and (t2, e2') = expr e2 in
        (* All binary operators require operands of the same type *)
        let same = t1 = t2 in
        let error = "illegal binary operator " ^
        string_of_typ t1 ^ " " ^ string_of_op op ^ " " ^
        string_of_typ t2 ^ " in " ^ string_of_expr e in
        (* Determine expression type based on operator and operand types *)
        let ty = match op with
          Add | Sub | Mult | Div when same && t1 = Int   -> Int
```

```
          | Add | Sub | Mult | Div when same && t1 = Float -> Float
          | Add | Sub | Elmult| Eldiv -> (match t1, t2 with
               Matrix(s1,a1,b1), Matrix(s2,a2,b2) ->
                 if s1=s2 && a1 = a2 && b1 = b2 then Matrix(s1,a1,b1)
                 else raise (Failure "illegal binary operator for matrix of
different sizes")
                  | _ -> raise (Failure error))
          | Mult  -> (match t1, t2 with
               Matrix(s1,a1,b1), Matrix(s2,a2,b2) ->
                 if s1=s2 && b1 = a2 then Matrix(s1,a1,b2)
                 else raise (Failure "illegal dimensions for matrix mult")
                 | _ -> raise (Failure error))
          | Equal | Neq            when same               -> Bool
          | Less | Leq | Greater | Geq
                     when same && (t1 = Int || t1 = Float) -> Bool
          | And | Or when same && t1 = Bool -> Bool
          | _ -> raise (Failure error)
          in (ty, SBinop((t1, e1'), op, (t2, e2')))
      | Call(fname, args) as call ->
          let fd = find_func fname in
          let param_length = List.length fd.formals in
          if List.length args != param_length then
            raise (Failure ("expecting " ^ string_of_int param_length ^
                            " arguments in " ^ string_of_expr call))
          else let check_call (ft, _) e =
            let (et, e') = expr e in
            let err = "illegal argument found " ^ string_of_typ et ^
              " expected " ^ string_of_typ ft ^ " in " ^ string_of_expr e
            in (check_assign ft et err, e')
          in
          let formals = List.map (fun (tp, var) -> (tp,var)) fd.formals in
          let args' = List.map2 check_call formals args
          in (fd.typ, SCall(fname, args'))

      | Mat(arr) ->
        let sArr = (List.map (fun l -> (List.map expr l)) arr) in
          let row_lengths = List.map List.length arr in
          if not (List.for_all (fun l -> if List.hd(row_lengths) = l then true else
false) row_lengths) then
            raise (Failure ("Matrix rows must be of same length"))
          else
            let (wt, _) = expr (List.hd(List.hd(arr))) in
            let r = List.length arr in
            let c = List.length (List.hd arr) in
            let expr_check = function
                (Int, _) when wt = Int -> true
              | (Float, _) when wt = Float -> true
```

```
              | _ -> raise (Failure("Matrix types don't match"))
            in
            ignore(List.for_all (fun j -> List.for_all (fun k -> expr_check (expr
k)) j) arr);
            (Matrix (wt, r, c), SMat(wt, sArr))
      | Col(s)     ->
          (match type_of_identifier s with
            Matrix(_,_,c) ->
              (match c with
              c -> (Int, SCol(c))
              | _ -> raise(Failure "Add int column value to matrix decl"))
            |_ -> raise(Failure "Cannot find column value of non-matrix"))
      | Row(s)     ->
          (match type_of_identifier s with
            Matrix(_,r,_) ->
              (match r with
              r -> (Int, SRow(r))
              | _ -> raise(Failure "Add int column value to matrix decl"))
            |_ -> raise(Failure "Cannot find column value of non-matrix"))
      | Tran(s)    ->
          (match type_of_identifier s with
            Matrix(t,r,c) -> (Matrix(t,c,r), STran(s, Matrix(t,r,c)))
            |_ -> raise(Failure "Cannot find column value of non-matrix"))
      | Access(s, r, c) -> let (row, row') = expr r in
                      let(col,col') = expr c in
                      if (col = Int)
                        then (if(row = Int)
                                then ()
                                else raise(Failure "row value is non-integer");)
                        else raise(Failure "column value is non-integer");
                      (match type_of_identifier s with
                      Matrix(t,_,_) -> (t, SAccess(s, (row, row'), (col,col')))
                      | _ -> raise(Failure "Cannot perform access operation on a
non-matrix type")
                      )

    in

    let check_bool_expr e =
      let (t', e') = expr e
      and err = "expected Boolean expression in " ^ string_of_expr e
      in if t' != Bool then raise (Failure err) else (t', e')
    in

    (* Return a semantically-checked statement i.e. containing sexprs *)
    let rec check_stmt = function
        Expr e -> SExpr (expr e)
```

```
      | If(p, b1, b2) -> SIf(check_bool_expr p, check_stmt b1, check_stmt b2)
      | For(e1, e2, e3, st) ->
        SFor(expr e1, check_bool_expr e2, expr e3, check_stmt st)
      | While(p, s) -> SWhile(check_bool_expr p, check_stmt s)
      | Return e -> let (t, e') = expr e in
        if t = func.typ then SReturn (t, e')
        else raise (
        Failure ("return gives " ^ string_of_typ t ^ " expected " ^
                string_of_typ func.typ ^ " in " ^ string_of_expr e))

          (* A block is correct if each statement is correct and nothing
             follows any Return statement.  Nested blocks are flattened. *)
      | Block sl ->
          let rec check_stmt_list = function
              [Return _ as s] -> [check_stmt s]
            | Return _ :: _   -> raise (Failure "nothing may follow a return")
            | Block sl :: ss  -> check_stmt_list (sl @ ss) (* Flatten blocks *)
            | s :: ss         -> check_stmt s :: check_stmt_list ss
            | []              -> []
          in SBlock(check_stmt_list sl)

    in (* body of check_function *)
    { styp = func.typ;
      sfname = func.fname;
      sformals = func.formals;
      slocals  = func.locals;
      sbody = match check_stmt (Block func.body) with
        SBlock(sl) -> sl
      | _ -> raise (Failure ("internal error: block didn't become a block?"))
    }
  in (globals, List.map check_function functions)
```

8.1.6 sast.ml

```
(* Semantically-checked Abstract Syntax Tree and functions for printing it
*)

open Ast

type sexpr = typ * sx
and sx =
    SLiteral of int
  | SFliteral of string
  | SBoolLit of bool
```

```
  | SStrLit of string
  | SId of string
  | SBinop of sexpr * op * sexpr
  | SUnop of uop * sexpr
  | SAssign of sexpr * sexpr
  | SCall of string * sexpr list
  | SMat of typ * sexpr list list
  | SCol of int
  | SRow of int
  | STran of string * typ
  | SAccess of string * sexpr * sexpr
  | SNoexpr


type sstmt =
    SBlock of sstmt list
  | SExpr of sexpr
  | SReturn of sexpr
  | SIf of sexpr * sstmt * sstmt
  | SFor of sexpr * sexpr * sexpr * sstmt
  | SWhile of sexpr * sstmt

type sfunc_decl = {
    styp : typ;
    sfname : string;
    sformals : bind list;
    slocals : bind list;
    sbody : sstmt list;
  }

type sprogram = bind list * sfunc_decl list

(* Pretty-printing functions *)

let rec string_of_sexpr (t, e) =
  "(" ^ string_of_typ t ^ " : " ^ (match e with
    SLiteral(l) -> string_of_int l
  | SBoolLit(true) -> "true"
  | SBoolLit(false) -> "false"
  | SStrLit(l) -> "\"" ^ (String.escaped l) ^ "\""
  | SFliteral(l) -> l
  | SMat(_,_) -> "matLit"
  | SId(s) -> s
```

```
  | SCol(s) -> (string_of_int s) ^ " columns"
  | SRow(s) -> (string_of_int s) ^ " rows"
  | STran (s,b) -> string_of_typ b ^ " " ^ s ^ ".T"
  | SAccess (s,e1,e2) -> s ^ "[" ^ string_of_sexpr e1 ^ "]" ^ "[" ^
string_of_sexpr e2 ^ "]"
  | SBinop(e1, o, e2) ->
      string_of_sexpr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_sexpr e2
  | SUnop(o, e) -> string_of_uop o ^ string_of_sexpr e
  | SAssign(v, e) -> string_of_sexpr v ^ " = " ^ string_of_sexpr e
  | SCall(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_sexpr el) ^ ")"
  | SNoexpr -> ""
                          ) ^ ")"


let rec string_of_sstmt = function
    SBlock(stmts) ->
      "{\n" ^ String.concat "" (List.map string_of_sstmt stmts) ^ "}\n"
  | SExpr(expr) -> string_of_sexpr expr ^ ";\n";
  | SReturn(expr) -> "return " ^ string_of_sexpr expr ^ ";\n";
  | SIf(e, s, SBlock([])) ->
      "if (" ^ string_of_sexpr e ^ ")\n" ^ string_of_sstmt s
  | SIf(e, s1, s2) ->  "if (" ^ string_of_sexpr e ^ ")\n" ^
      string_of_sstmt s1 ^ "else\n" ^ string_of_sstmt s2
  | SFor(e1, e2, e3, s) ->
      "for (" ^ string_of_sexpr e1  ^ " ; " ^ string_of_sexpr e2 ^ " ; " ^
      string_of_sexpr e3  ^ ") " ^ string_of_sstmt s
  | SWhile(e, s) -> "while (" ^ string_of_sexpr e ^ ") " ^ string_of_sstmt
s

let string_of_sfdecl fdecl =
  string_of_typ fdecl.styp ^ " " ^
  fdecl.sfname ^ "(" ^ String.concat ", " (List.map snd fdecl.sformals) ^
  ")\n{\n" ^
  String.concat "" (List.map string_of_vdecl fdecl.slocals) ^
  String.concat "" (List.map string_of_sstmt fdecl.sbody) ^
  "}\n"

let string_of_sprogram (vars, funcs) =
  String.concat "" (List.map string_of_vdecl vars) ^ "\n" ^
  String.concat "\n" (List.map string_of_sfdecl funcs)
```

8.1.7 codegen.ml

```
(* Code generation: translate takes a semantically checked AST and
produces LLVM IR

LLVM tutorial: Make sure to read the OCaml version of the tutorial

http://llvm.org/docs/tutorial/index.html

Detailed documentation on the OCaml LLVM library:

http://llvm.moe/
http://llvm.moe/ocaml/

*)

module L = Llvm
module A = Ast
open Sast

module StringMap = Map.Make(String)

(* translate : Sast.program -> Llvm.module *)
let translate (globals, functions) =
  let context    = L.global_context () in

  (* Create the LLVM compilation module into which
     we will generate code *)
  let the_module = L.create_module context "redpandas" in

  (* Get types from the context *)
  let i32_t      = L.i32_type    context
  and i8_t       = L.i8_type     context
  and i1_t       = L.i1_type     context
  and float_t    = L.double_type context
  and void_t     = L.void_type   context
  and pointer_t  = L.pointer_type
  and array_t    = L.array_type
  in

  (* Return the LLVM type for a redpandas type *)
  let ltype_of_typ = function
```

```
    A.Int   -> i32_t
  | A.Bool  -> i1_t
  | A.Float -> float_t
  | A.Void  -> void_t
  | A.String -> pointer_t i8_t
  | A.Matrix(t,r,c) ->
      let rows = match r with s -> s
        | _ -> raise(Failure"Integer required for matrix dimension") in
      let cols = match c with s -> s
        | _ -> raise(Failure"Integer required for matrix dimension") in
        (match t with
          A.Int    -> array_t (array_t i32_t cols) rows
        | A.Float  -> array_t (array_t float_t cols) rows
        | _        -> raise(Failure"Invalid datatype for matrix"))
in

(* Create a map of global variables after creating each *)
let global_vars : L.llvalue StringMap.t =
  let global_var m (t, n) =
    let init = match t with
        A.Float -> L.const_float (ltype_of_typ t) 0.0
      | _ -> L.const_int (ltype_of_typ t) 0
    in StringMap.add n (L.define_global n init the_module) m in
  List.fold_left global_var StringMap.empty globals in

let printf_t : L.lltype =
    L.var_arg_function_type i32_t [| L.pointer_type i8_t |] in
let printf_func : L.llvalue =
    L.declare_function "printf" printf_t the_module in

let printbig_t : L.lltype =
    L.function_type i32_t [| i32_t |] in
let printbig_func : L.llvalue =
    L.declare_function "printbig" printbig_t the_module in

(* Define each function (arguments and return type) so we can
   call it even before we've created its body *)
let function_decls : (L.llvalue * sfunc_decl) StringMap.t =
  let function_decl m fdecl =
    let name = fdecl.sfname
    and formal_types =
    Array.of_list (List.map (fun (t,_) -> ltype_of_typ t) fdecl.sformals)
    in let ftype = L.function_type (ltype_of_typ fdecl.styp) formal_types
```

```
in
      StringMap.add name (L.define_function name ftype the_module, fdecl) m
in
    List.fold_left function_decl StringMap.empty functions in

  (* Fill in the body of the given function *)
  let build_function_body fdecl =
    let (the_function, _) = StringMap.find fdecl.sfname function_decls in
    let builder = L.builder_at_end context (L.entry_block the_function) in

    let int_format_str = L.build_global_stringptr "%d\t" "fmt" builder
    and string_format_str =  L.build_global_stringptr "%s\n" "fmt" builder
    and float_format_str = L.build_global_stringptr "%g\t" "fmt" builder in

    (* Construct the function's "locals": formal arguments and locally
       declared variables.  Allocate each on the stack, initialize their
       value, if appropriate, and remember their values in the "locals" map
*)
    let local_vars =
      let add_formal m (t, n) p =
        L.set_value_name n p;
      let local = L.build_alloca (ltype_of_typ t) n builder in
        ignore (L.build_store p local builder);
      StringMap.add n local m

      (* Allocate space for any locally declared variables and add the
       * resulting registers to our map *)
      and add_local m (t, n) =
      let local_var = L.build_alloca (ltype_of_typ t) n builder
      in StringMap.add n local_var m
      in

      let formals = List.fold_left2 add_formal StringMap.empty
fdecl.sformals
          (Array.to_list (L.params the_function)) in
      List.fold_left add_local formals fdecl.slocals
    in

    (* Return the value for a variable or formal argument.
       Check local names first, then global names *)
    let lookup n = try StringMap.find n local_vars
                   with Not_found -> StringMap.find n global_vars
    in
```

```
    let accessValue s r c builder a =
      let specific = L.build_gep (lookup s) [|L.const_int i32_t 0; r; c|] s
builder in
      if a then specific else L.build_load specific s builder
    in

    (* Construct code for an expression; return its value *)
    let rec expr builder ((_, e) : sexpr) = match e with
          SLiteral i  -> L.const_int i32_t i
      | SBoolLit b  -> L.const_int i1_t (if b then 1 else 0)
      | SFliteral l -> L.const_float_of_string float_t l
      | SStrLit s   -> L.build_global_stringptr s "tmp" builder
      | SNoexpr     -> L.const_int i32_t 0
      | SId s       -> L.build_load (lookup s) s builder
      | SMat (t, mat) ->
        let innertype = match t with
                A.Float -> float_t
                | A.Int -> i32_t
                | _ -> i32_t
          in
            let lists       = List.map (List.map (expr builder)) mat in
            let innerArray   = List.map Array.of_list lists in
            let list2array  = Array.of_list ((List.map (L.const_array
innertype) innerArray)) in
            L.const_array (array_t innertype (List.length (List.hd mat)))
list2array
      | SCol (c)   -> L.const_int i32_t c
      | SRow (r)   -> L.const_int i32_t r
      | STran (s,t)  ->
                    let typ = match t with
                    Matrix(Int, _, _) -> i32_t | Matrix(Float, _, _) ->
float_t| _ -> i32_t in
                    (match t with
                    Matrix(Int, c, r) | Matrix(Float, c, r) ->
                        let tempAlloc = L.build_alloca (array_t (array_t
typ c) r) "tmpmat" builder in
                        for i=0 to (c-1) do
                            for j=0 to (r-1) do
                                let temp = accessValue s (L.const_int i32_t
i) (L.const_int i32_t j) builder false in
                                let l = L.build_gep tempAlloc [|
L.const_int i32_t 0; L.const_int i32_t j; L.const_int i32_t i |] "tmpmat"
```

```
builder in
                                ignore(L.build_store temp l builder);
                              done
                          done;
                          L.build_load (L.build_gep tempAlloc [| L.const_int
i32_t 0 |] "tmpmat" builder) "tmpmat" builder
                      | _ -> L.const_int i32_t 0)
      | SAccess (s,r,c) -> let a = expr builder r and b = expr builder c in
                           (accessValue s a b builder false)
      | SAssign (s, e) -> let e' = expr builder e and
                           s' = (match s with
                             (_, SAccess(t,r,c)) -> let a = expr builder r
and b = expr builder c in
                               accessValue t a b builder true
                           | (_,SId(t)) -> lookup t
                           | _      -> raise(Failure "Value is not
assignable")) in
                           ignore(L.build_store e' s' builder); e'
      | SBinop ((A.Float,_ ) as e1, op, e2) ->
          let e1' = expr builder e1
          and e2' = expr builder e2 in
          (match op with
            A.Add     -> L.build_fadd
          | A.Sub     -> L.build_fsub
          | A.Mult    -> L.build_fmul
          | A.Div     -> L.build_fdiv
          | A.Equal   -> L.build_fcmp L.Fcmp.Oeq
          | A.Neq     -> L.build_fcmp L.Fcmp.One
          | A.Less    -> L.build_fcmp L.Fcmp.Olt
          | A.Leq     -> L.build_fcmp L.Fcmp.Ole
          | A.Greater -> L.build_fcmp L.Fcmp.Ogt
          | A.Geq     -> L.build_fcmp L.Fcmp.Oge
          | A.And | A.Or ->
              raise (Failure "internal error: semant should have rejected
and/or on float")
          | _ ->  raise (Failure "error: not a viable int to int
operation")
          ) e1' e2' "tmp" builder
      | SBinop (e1, op, e2) ->
          let e1' = expr builder e1
          and e2' = expr builder e2
          and (typ1,_) = e1
          and (typ2,_) = e2  in
```

```
          let str1 = (match e1 with (_, SId(s)) -> s | _ -> "") in
          let str2 = (match e2 with (_, SId(s)) -> s | _ -> "") in
          (match (typ1, typ2) with
          | (Int, Int)   ->  (match op with
                                A.Add     -> L.build_add
                              | A.Sub     -> L.build_sub
                              | A.Mult    -> L.build_mul
                                  | A.Div     -> L.build_sdiv
                              | A.And     -> L.build_and
                              | A.Or      -> L.build_or
                              | A.Equal   -> L.build_icmp L.Icmp.Eq
                              | A.Neq     -> L.build_icmp L.Icmp.Ne
                              | A.Less    -> L.build_icmp L.Icmp.Slt
                              | A.Leq     -> L.build_icmp L.Icmp.Sle
                              | A.Greater -> L.build_icmp L.Icmp.Sgt
                              | A.Geq     -> L.build_icmp L.Icmp.Sge
                              | _ ->  raise (Failure "error: not a viable
int to int operation")    ) e1' e2' "tmp" builder
          | (Matrix(Int, a1, b1), Matrix(Int, _, b2)) ->
                                (match op with
                              | A.Add ->
                                  let temp = L.build_alloca (array_t
(array_t i32_t b2) a1) "tmpmat" builder in
                                  for i = 0 to (a1-1) do
                                    for j = 0 to (b2-1) do
                                      let mat1 = accessValue str1
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                      let mat2 = accessValue str2
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                      let final = L.build_add mat1 mat2
"tmp" builder in
                                      let l = L.build_gep temp [|
L.const_int i32_t 0; L.const_int i32_t i; L.const_int i32_t j |] "tmpmat"
builder in
                                      ignore(L.build_store final l
builder);
                                    done
                                  done;
                                  L.build_load (L.build_gep temp [|
L.const_int i32_t 0 |] "tmpmat" builder) "tmpmat" builder

                              | A.Sub ->
                                let temp = L.build_alloca (array_t
```

```
(array_t i32_t b2) a1) "tmpmat" builder in
                                  for i = 0 to (a1-1) do
                                    for j = 0 to (b2-1) do
                                      let mat1 = accessValue str1
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                      let mat2 = accessValue str2
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                      let final = L.build_sub mat1 mat2
"tmp" builder in
                                      let l = L.build_gep temp [|
L.const_int i32_t 0; L.const_int i32_t i; L.const_int i32_t j |] "tmpmat"
builder in
                                      ignore(L.build_store final l
builder);
                                    done
                                  done;
                                  L.build_load (L.build_gep temp [|
L.const_int i32_t 0 |] "tmpmat" builder) "tmpmat" builder

                                | A.Elmult ->
                                  let temp = L.build_alloca (array_t
(array_t i32_t b2) a1) "tmpmat" builder in
                                  for i = 0 to (a1-1) do
                                    for j = 0 to (b2-1) do
                                      let mat1 = accessValue str1
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                      let mat2 = accessValue str2
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                      let final = L.build_mul mat1 mat2
"tmp" builder in
                                      let l = L.build_gep temp [|
L.const_int i32_t 0; L.const_int i32_t i; L.const_int i32_t j |] "tmpmat"
builder in
                                      ignore(L.build_store final l
builder);
                                    done
                                  done;
                                  L.build_load (L.build_gep temp [|
L.const_int i32_t 0 |] "tmpmat" builder) "tmpmat" builder
                                | A.Eldiv ->
                                  let temp = L.build_alloca (array_t
(array_t i32_t b2) a1) "tmpmat" builder in
                                  for i = 0 to (a1-1) do
```

```
                                    for j = 0 to (b2-1) do
                                      let mat1 = accessValue str1
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                      let mat2 = accessValue str2
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                      let final = L.build_sdiv mat1 mat2
"tmp" builder in
                                      let l = L.build_gep temp [|
L.const_int i32_t 0; L.const_int i32_t i; L.const_int i32_t j |] "tmpmat"
builder in
                                      ignore(L.build_store final l
builder);
                                    done
                                  done;
                                  L.build_load (L.build_gep temp [|
L.const_int i32_t 0 |] "tmpmat" builder) "tmpmat" builder

                                | A.Mult ->
                                  let temp = L.build_alloca (array_t
(array_t i32_t b2) a1) "tmpmat" builder in
                                  let temp_val = L.build_alloca i32_t
"tmpval" builder in
                                  ignore (L.build_store (L.const_int i32_t
0) temp_val builder);
                                  for i = 0 to (a1-1) do
                                    for j = 0 to (b2-1) do
                                      ignore (L.build_store (L.const_int
i32_t 0) temp_val builder);
                                      for k = 0 to (b1-1) do
                                        let mat1 = accessValue str1
(L.const_int i32_t i) (L.const_int i32_t k) builder false in
                                        let mat2 = accessValue str2
(L.const_int i32_t k) (L.const_int i32_t j) builder false in
                                        let final = L.build_mul mat1 mat2
"tmp" builder in
                                        ignore(L.build_store (L.build_add
final (L.build_load temp_val "addtmp" builder) "tmp" builder) temp_val
builder);
                                      done;
                                      let l = L.build_gep temp [|
L.const_int i32_t 0; L.const_int i32_t i; L.const_int i32_t j |] "tmpmat"
builder in
                                      ignore(L.build_store (L.build_load
```

```
temp_val "restmp" builder) l builder);
                                    done
                                  done;
                                  L.build_load (L.build_gep temp [|
L.const_int i32_t 0 |] "tmpmat" builder) "tmpmat" builder
                                  | _ ->  raise (Failure "error: not a
viable matrix to matrix operation")
                                  )
            | (Matrix(Float, a1, b1), Matrix(Float, _, b2)) ->
                                  (match op with
                                  | A.Add ->
                                      let temp = L.build_alloca (array_t
(array_t float_t b2) a1) "tmpmat" builder in
                                      for i = 0 to (a1-1) do
                                        for j = 0 to (b2-1) do
                                          let mat1 = accessValue str1
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                          let mat2 = accessValue str2
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                          let final = L.build_fadd mat1
mat2 "tmp" builder in
                                          let l = L.build_gep temp [|
L.const_int i32_t 0; L.const_int i32_t i; L.const_int i32_t j |] "tmpmat"
builder in
                                          ignore(L.build_store final l
builder);
                                        done
                                      done;
                                      L.build_load (L.build_gep temp [|
L.const_int i32_t 0 |] "tmpmat" builder) "tmpmat" builder

                                  | A.Sub ->
                                    let temp = L.build_alloca (array_t
(array_t float_t b2) a1) "tmpmat" builder in
                                    for i = 0 to (a1-1) do
                                      for j = 0 to (b2-1) do
                                        let mat1 = accessValue str1
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                        let mat2 = accessValue str2
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                        let final = L.build_fsub mat1 mat2
"tmp" builder in
                                        let l = L.build_gep temp [|
```

```
L.const_int i32_t 0; L.const_int i32_t i; L.const_int i32_t j |] "tmpmat"
builder in
                                        ignore(L.build_store final l
builder);
                                      done
                                    done;
                                  L.build_load (L.build_gep temp [|
L.const_int i32_t 0 |] "tmpmat" builder) "tmpmat" builder
                                  | A.Elmult ->
                                    let temp = L.build_alloca (array_t
(array_t float_t b2) a1) "tmpmat" builder in
                                    for i = 0 to (a1-1) do
                                      for j = 0 to (b2-1) do
                                        let mat1 = accessValue str1
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                        let mat2 = accessValue str2
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                        let final = L.build_fmul mat1 mat2
"tmp" builder in
                                        let l = L.build_gep temp [|
L.const_int i32_t 0; L.const_int i32_t i; L.const_int i32_t j |] "tmpmat"
builder in
                                        ignore(L.build_store final l
builder);
                                      done
                                    done;
                                    L.build_load (L.build_gep temp [|
L.const_int i32_t 0 |] "tmpmat" builder) "tmpmat" builder

                                | A.Eldiv ->
                                  let temp = L.build_alloca (array_t
(array_t float_t b2) a1) "tmpmat" builder in
                                  for i = 0 to (a1-1) do
                                    for j = 0 to (b2-1) do
                                      let mat1 = accessValue str1
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                      let mat2 = accessValue str2
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                      let final = L.build_fdiv mat1 mat2
"tmp" builder in
                                      let l = L.build_gep temp [|
L.const_int i32_t 0; L.const_int i32_t i; L.const_int i32_t j |] "tmpmat"
builder in
```

```
                                    ignore(L.build_store final l
builder);
                                  done
                                done;
                                L.build_load (L.build_gep temp [|
L.const_int i32_t 0 |] "tmpmat" builder) "tmpmat" builder
                                | A.Mult ->
                                  let temp = L.build_alloca (array_t
(array_t i32_t b2) a1) "tmpmat" builder in
                                  let temp_val = L.build_alloca float_t
"tmpval" builder in
                                  ignore (L.build_store (L.const_float
float_t 0.0) temp_val builder);
                                  for i = 0 to (a1-1) do
                                    for j = 0 to (b2-1) do
                                      ignore (L.build_store
(L.const_float float_t 0.0) temp_val builder);
                                      for k = 0 to (b1-1) do
                                        let mat1 = accessValue str1
(L.const_int i32_t i) (L.const_int i32_t k) builder false in
                                        let mat2 = accessValue str2
(L.const_int i32_t k) (L.const_int i32_t j) builder false in
                                        let final = L.build_fmul mat1
mat2 "tmp" builder in
                                        ignore(L.build_store
(L.build_fadd final (L.build_load temp_val "addtmp" builder) "tmp" builder)
temp_val builder);
                                      done;
                                      let l = L.build_gep temp [|
L.const_int i32_t 0; L.const_int i32_t i; L.const_int i32_t j |] "tmpmat"
builder in
                                      ignore(L.build_store (L.build_load
temp_val "restmp" builder) l builder);
                                    done
                                  done;
                                  L.build_load (L.build_gep temp [|
L.const_int i32_t 0 |] "tmpmat" builder) "tmpmat" builder
                                | _ ->  raise (Failure "error: not a
viable matrix to matrix operation")
                                )
          | _ -> (match op with
                    A.Add     -> L.build_add
                  | A.Sub     -> L.build_sub
```

```
                    | A.Mult    -> L.build_mul
                         | A.Div     -> L.build_sdiv
                    | A.And     -> L.build_and
                    | A.Or      -> L.build_or
                    | A.Equal   -> L.build_icmp L.Icmp.Eq
                    | A.Neq     -> L.build_icmp L.Icmp.Ne
                    | A.Less    -> L.build_icmp L.Icmp.Slt
                    | A.Leq     -> L.build_icmp L.Icmp.Sle
                    | A.Greater -> L.build_icmp L.Icmp.Sgt
                    | A.Geq     -> L.build_icmp L.Icmp.Sge
                    | _ ->  raise (Failure "error: not a viable operation")
) e1' e2' "tmp" builder
        )

      | SUnop(op, ((t, _) as e)) ->
          let e' = expr builder e in
        (match op with
          A.Neg when t = A.Float -> L.build_fneg
        | A.Neg                  -> L.build_neg
          | A.Not                  -> L.build_not) e' "tmp" builder
      | SCall ("print", [e]) | SCall ("printb", [e]) ->
            L.build_call printf_func [| int_format_str ; (expr builder e)
|]
          "printf" builder
      | SCall ("printbig", [e]) ->
            L.build_call printbig_func [| (expr builder e) |] "printbig"
builder
      | SCall ("printf", [e]) ->
            L.build_call printf_func [| float_format_str ; (expr builder e)
|]
          "printf" builder
      | SCall ("printStr", [e]) ->
        L.build_call printf_func [| string_format_str ; (expr builder e) |]
        "printf" builder
      | SCall (f, args) ->
         let (fdef, fdecl) = StringMap.find f function_decls in
       let llargs = List.rev (List.map (expr builder) (List.rev args)) in
       let result = (match fdecl.styp with
                        A.Void -> ""
                      | _ -> f ^ "_result") in
         L.build_call fdef (Array.of_list llargs) result builder
    in
```

```
    (* LLVM insists each basic block end with exactly one "terminator"
       instruction that transfers control.  This function runs "instr
builder"
       if the current block does not already have a terminator.  Used,
       e.g., to handle the "fall off the end of the function" case. *)
    let add_terminal builder instr =
      match L.block_terminator (L.insertion_block builder) with
      Some _ -> ()
      | None -> ignore (instr builder) in

    (* Build the code for the given statement; return the builder for
       the statement's successor (i.e., the next instruction will be built
       after the one generated by this call) *)

    let rec stmt builder = function
          SBlock sl -> List.fold_left stmt builder sl
      | SExpr e -> ignore(expr builder e); builder
      | SReturn e -> ignore(match fdecl.styp with
                              (* Special "return nothing" instr *)
                              A.Void -> L.build_ret_void builder
                              (* Build return statement *)
                            | _ -> L.build_ret (expr builder e) builder );
                     builder
      | SIf (predicate, then_stmt, else_stmt) ->
        let bool_val = expr builder predicate in
            let merge_bb = L.append_block context "merge" the_function in
        let build_br_merge = L.build_br merge_bb in (* partial function *)

       let then_bb = L.append_block context "then" the_function in
       add_terminal (stmt (L.builder_at_end context then_bb) then_stmt)
         build_br_merge;

       let else_bb = L.append_block context "else" the_function in
       add_terminal (stmt (L.builder_at_end context else_bb) else_stmt)
         build_br_merge;

       ignore(L.build_cond_br bool_val then_bb else_bb builder);
       L.builder_at_end context merge_bb

      | SWhile (predicate, body) ->
        let pred_bb = L.append_block context "while" the_function in
        ignore(L.build_br pred_bb builder);
```

```
        let body_bb = L.append_block context "while_body" the_function in
        add_terminal (stmt (L.builder_at_end context body_bb) body)
          (L.build_br pred_bb);

        let pred_builder = L.builder_at_end context pred_bb in
        let bool_val = expr pred_builder predicate in

        let merge_bb = L.append_block context "merge" the_function in
        ignore(L.build_cond_br bool_val body_bb merge_bb pred_builder);
        L.builder_at_end context merge_bb

      (* Implement for loops as while loops *)
      | SFor (e1, e2, e3, body) -> stmt builder
          ( SBlock [SExpr e1 ; SWhile (e2, SBlock [body ; SExpr e3]) ] )
    in

    (* Build the code for each statement in the function *)
    let builder = stmt builder (SBlock fdecl.sbody) in

    (* Add a return if the last block falls off the end *)
    add_terminal builder (match fdecl.styp with
        A.Void -> L.build_ret_void
      | A.Float -> L.build_ret (L.const_float float_t 0.0)
      | t -> L.build_ret (L.const_int (ltype_of_typ t) 0))
  in

  List.iter build_function_body functions;
  the_module
```

8.1.8 redpandas.ml

```
type action = Ast | Sast| LLVM_IR | Compile

let _ =
  let action = ref Compile in
  let set_action a () = action := a in
  let speclist = [
    ("-a", Arg.Unit (set_action Ast), "Print the SAST");
    ("-l", Arg.Unit (set_action LLVM_IR), "Print the generated LLVM IR");
    ("-c", Arg.Unit (set_action Compile),
      "Check and print the generated LLVM IR (default)");
  ] in
```

```
  let usage_msg = "usage: ./redpandas.native [-a|-l|-c] [file.rp]" in
  let channel = ref stdin in
  Arg.parse speclist (fun filename -> channel := open_in filename)
usage_msg;

  let lexbuf = Lexing.from_channel !channel in
  let ast = Parser.program Scanner.token lexbuf in
  match !action with
    Ast -> print_string (Ast.string_of_program ast)
  | _ -> let sast = Semant.check ast in
    match !action with
      Ast     -> ()
    | Sast    -> print_string (Sast.string_of_sprogram sast)
    | LLVM_IR -> print_string (Llvm.string_of_llmodule (Codegen.translate
sast))
    | Compile -> let m = Codegen.translate sast in
      Llvm_analysis.assert_valid_module m;
      print_string (Llvm.string_of_llmodule m)
```

8.1.9 rpc

```
#!/bin/bash
redpandas="./redpandas.native"

error=0
globalerror=0
keep=0

SignalError() {
    if [ $error -eq 0 ] ; then
      echo "FAILED"
      error=1
    fi
    echo "  $1"
}

# Run <args>
# Report the command, run it, report and exit on any errors
Run() {
    echo $* 1>&2
    eval $*
```

```
}

Check() {
    error=0
    b=`echo $1 | sed 's/.*\\///
                      s/.rp//'`

Run "$redpandas" "$1" ">" "${b}.ll" &&
    Run "llc" "-relocation-model=pic" "${b}.ll" ">" "${b}.s" &&
    Run "cc" "-o" "${b}.exe" "${b}.s" "printbig.o" &&
    Run "rm" "*.ll" "*.s" &&
    Run "./${b}.exe"

}

if [ $# -lt 1 ]
then
    echo "Usage: rpc [.rp file]"
    exit 1
fi

Check $1

exit $globalerror
```

8.1.10 testall.sh

```
#!/bin/sh

# Regression testing script for redpandas
# Step through a list of files
#  Compile, run, and check the output of each expected-to-work test
#  Compile and check the error of each expected-to-fail test

# Path to the LLVM interpreter
LLI="lli"
#LLI="/usr/local/opt/llvm/bin/lli"
```

```
# Path to the LLVM compiler
LLC="llc"

# Path to the C compiler
CC="cc"

# Path to the redpandas compiler.  Usually "./redpandas.native"
# Try "_build/redpandas.native" if ocamlbuild was unable to create a
symbolic link.
redpandas="./redpandas.native"
#redpandas="_build/redpandas.native"

# Set time limit for all operations
ulimit -t 30

globallog=testall.log
rm -f $globallog
error=0
globalerror=0

keep=0

Usage() {
    echo "Usage: testall.sh [options] [.rp files]"
    echo "-k    Keep intermediate files"
    echo "-h    Print this help"
    exit 1
}

SignalError() {
    if [ $error -eq 0 ] ; then
      echo "FAILED"
      error=1
    fi
    echo "  $1"
}

# Compare <outfile> <reffile> <difffile>
# Compares the outfile with reffile.  Differences, if any, written to
```

```
difffile
Compare() {
    generatedfiles="$generatedfiles $3"
    echo diff -b $1 $2 ">" $3 1>&2
    diff -b "$1" "$2" > "$3" 2>&1 || {
      SignalError "$1 differs"
      echo "FAILED $1 differs from $2" 1>&2
    }
}

# Run <args>
# Report the command, run it, and report any errors
Run() {
    echo $* 1>&2
    eval $* || {
      SignalError "$1 failed on $*"
      return 1
    }
}

# RunFail <args>
# Report the command, run it, and expect an error
RunFail() {
    echo $* 1>&2
    eval $* && {
      SignalError "failed: $* did not report an error"
      return 1
    }
    return 0
}

Check() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
                              s/.rp//'`
    reffile=`echo $1 | sed 's/.rp$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."
```

```
    echo 1>&2
    echo "###### Testing $basename" 1>&2

    generatedfiles=""

    generatedfiles="$generatedfiles ${basename}.ll ${basename}.s
${basename}.exe ${basename}.out" &&
    Run "$redpandas" "$1" ">" "${basename}.ll" &&
    Run "$LLC" "-relocation-model=pic" "${basename}.ll" ">"
"${basename}.s" &&
    Run "$CC" "-o" "${basename}.exe" "${basename}.s" "printbig.o" &&
    Run "./${basename}.exe" > "${basename}.out" &&
    Compare ${basename}.out ${reffile}.out ${basename}.diff

    # Report the status and clean up the generated files

    if [ $error -eq 0 ] ; then
      if [ $keep -eq 0 ] ; then
          rm -f $generatedfiles
      fi
      echo "OK"
      echo "###### SUCCESS" 1>&2
    else
      echo "###### FAILED" 1>&2
      globalerror=$error
    fi
}

CheckFail() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
                              s/.rp//'`
    reffile=`echo $1 | sed 's/.rp$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

    echo 1>&2
```

```
    echo "###### Testing $basename" 1>&2

    generatedfiles=""

    generatedfiles="$generatedfiles ${basename}.err ${basename}.diff"
&&
    RunFail "$redpandas" "<" $1 "2>" "${basename}.err" ">>"
$globallog &&
    Compare ${basename}.err ${reffile}.err ${basename}.diff

    # Report the status and clean up the generated files

    if [ $error -eq 0 ] ; then
      if [ $keep -eq 0 ] ; then
          rm -f $generatedfiles
      fi
      echo "OK"
      echo "###### SUCCESS" 1>&2
    else
      echo "###### FAILED" 1>&2
      globalerror=$error
    fi
}

while getopts kdpsh c; do
    case $c in
      k) # Keep intermediate files
          keep=1
          ;;
      h) # Help
          Usage
          ;;
    esac
done

shift `expr $OPTIND - 1`

LLIFail() {
  echo "Could not find the LLVM interpreter \"$LLI\"."
```

```
  echo "Check your LLVM installation and/or modify the LLI variable
in testall.sh"
  exit 1
}

which "$LLI" >> $globallog || LLIFail

if [ ! -f printbig.o ]
then
    echo "Could not find printbig.o"
    echo "Try \"make printbig.o\""
    exit 1
fi

if [ $# -ge 1 ]
then
    files=$@
else
    files="tests/test-*.rp tests/fail-*.rp"
fi

for file in $files
do
    case $file in
     *test-*)
         Check $file 2>> $globallog
         ;;
     *fail-*)
         CheckFail $file 2>> $globallog
         ;;
     *)
         echo "unknown file type $file"
         globalerror=1
         ;;
    esac
done

exit $globalerror
```

8.2 Git Log

```
commit 4ba5cdf87e25e96fdf93c4d2315c5ba4d75afec2
Merge: b3f982c fd83515
Author: myric <18725865+myric@users.noreply.github.com>
Date:   Mon Apr 26 21:50:25 2021 +0100

    Merge pull request #5 from ibarral18/hello

    Hello

commit fd835153b0edd93bc437a7a1921478a07f24fed2
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Mon Apr 26 18:56:54 2021 +0100

    Additions to gitignore

commit ff12e013070fd1c92f8183a9c40b73598f333ac8
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Mon Apr 26 16:57:20 2021 +0100

    Removing Division tests

commit 0c46af096b36693b8f59ebdd5618caf600d2e1aa
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Mon Apr 26 16:45:09 2021 +0100

    Adjusted comparison file.

commit 9af4ff9ac619c828e6bbe4715c4fcaecfd588e76
Author: aassal <assal.amina.h@gmail.com>
Date:   Mon Apr 26 08:41:51 2021 -0700

    fixed err/out files

commit f4dea99dc91327d30bb0bcd8ddae270c67ffabd9
Author: aassal <assal.amina.h@gmail.com>
Date:   Mon Apr 26 08:22:10 2021 -0700

    added err files and failing tests for 3+ matrices

commit b3f982c41c319ba457832b1aeeb05f79de2db58d
```

```
Merge: 6f9fd28 eddea56
Author: myric <18725865+myric@users.noreply.github.com>
Date:   Mon Apr 26 16:02:36 2021 +0100

    Merge pull request #4 from ibarral18/hello

    Hello

commit eddea56b00eb02112a0d6f493131e6a213c4994f
Author: aassal <assal.amina.h@gmail.com>
Date:   Mon Apr 26 00:40:32 2021 -0700

    added access tests, fail test gives rando number not error

commit ebcf3a67d99a15c7172fefe307e1349e0bb25783
Author: aassal <assal.amina.h@gmail.com>
Date:   Mon Apr 26 00:34:00 2021 -0700

    added failing transpose test

commit a4d1ae39e0b02b481b15d61516195b5ebf2eaca8
Author: aassal <assal.amina.h@gmail.com>
Date:   Mon Apr 26 00:29:39 2021 -0700

    added transpose tests

commit 9249dc1422ad2c5d0f569de121eba8c25e58c77d
Author: aassal <assal.amina.h@gmail.com>
Date:   Mon Apr 26 00:13:01 2021 -0700

    added matrix test for add, mult, sub

commit 4d7ca7e2a9a5f087dd4fdb1880b83efd199e985e
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Sun Apr 25 12:23:21 2021 -0400

    fixing out files for tests

commit e37054574616d4fda45e480550cc1573c48b6f1d
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Sun Apr 25 16:56:11 2021 +0100

    Added basic demo programs
```

```
commit c1cfb48cdd3f45ce38484ed886cb495f83e1da3c
Author: aassal <assal.amina.h@gmail.com>
Date:   Sun Apr 25 08:36:17 2021 -0700

    demo with better printing

commit f7c6357a752f67ed102af2ff5ab565ded893b63b
Merge: 8013d5c 41fe022
Author: aassal <assal.amina.h@gmail.com>
Date:   Sun Apr 25 08:24:36 2021 -0700

    committing so i can switch branches

commit 8013d5c662104643a71ff8d93b00626b1db20dbc
Author: aassal <assal.amina.h@gmail.com>
Date:   Sun Apr 25 07:48:12 2021 -0700

    committing so i can pull new changes

commit 41fe022c331b23d1329ebf81e374cf29a464aad5
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Sun Apr 25 15:44:52 2021 +0100

    Fixed parsing errors on Demo

commit f55ab16ca929663b26b363d2a5db22b8fe85786a
Author: aassal <assal.amina.h@gmail.com>
Date:   Sun Apr 25 07:31:30 2021 -0700

    demo with parse error

commit 774fb3b094a6aeee533b434218231e274c960641
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Sun Apr 25 14:17:47 2021 +0100

    Adding gitignore and minor test updates

commit 55ee3f97bb5724f4655768b762156a8e9c3ded7b
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Sat Apr 24 20:42:34 2021 -0400

    fixing errors
```

```
commit 1e7775ec4a3b7161a5ee94f366db095f19308520
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Sun Apr 25 01:23:21 2021 +0100

    All tests redone with def

commit 97a7b9a1d030cdec36cea216cb3a1fb931fd7093
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Sun Apr 25 01:16:43 2021 +0100

    up to test-gcd2.rp

commit f9f792cfbcce0d643c24d8eaadee7a4bbbb20462
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Sat Apr 24 20:15:27 2021 -0400

    fixing some errors :/

commit 2426470f9d59c05d96417509881fcd0c9e006c4a
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Sun Apr 25 00:54:56 2021 +0100

    Down to fail-return2.rp

commit 977bfb6531fcfda7a14fd501b9f1eb7603343dbc
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Sun Apr 25 00:18:26 2021 +0100

    Fixed first set of tests

commit c6055c1534912778936d515807aa4660919e0d9a
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Sat Apr 24 17:30:49 2021 +0100

    Added 'make smallclean' into Makefile to help with cleaning up after
testing if you don't want to wipe everything."

commit 89621d180a24007adb2f07db5e434c5e38362715
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Fri Apr 23 21:53:48 2021 -0400

    adding transpose
```

```
commit 9cba0e24c932e8309210b0f32d0ea1e4457f5705
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Fri Apr 23 20:58:17 2021 -0400

    adding elementwise division/multiplication

commit b6a56e752ce15137b1e6e44d6750f5f8846b65f6
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Fri Apr 23 20:32:10 2021 -0400

    adding matrix multiplication and float functions

commit 96993c055b29a8cd99cf1cfa7289e784a30fe241
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Fri Apr 23 19:42:14 2021 -0400

    adding and subtracting matrices by each other

commit e02568e293082905588cfdade8675f410c0f97c8
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Fri Apr 23 17:12:58 2021 -0400

    adding easy file compile

commit a83151f3569d080cb99e72b865e70043ffa3df4c
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Fri Apr 23 01:55:59 2021 -0400

    adding new functionality to helloworld

commit 3b551d5a588e36c4d5b4723acb61340c9fb4afca
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Fri Apr 23 01:51:59 2021 -0400

    fixing assign to allow matrix index assigning

commit 0d86ae1220e42f2ac001f565c7398b7b271442fc6
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Thu Apr 22 23:57:31 2021 -0400

    adding matrix indexing
```

```
commit 1e9cb1394d85cdba0f349d27910c64c536fbdd77
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Thu Apr 22 18:11:57 2021 -0400

    adding column and row count

commit fd124dd8df9e8d844faa1498ae3804bf797cb554
Merge: 2b767a8 2e3cbd1
Author: Ivan Barral <51125955+ibarral18@users.noreply.github.com>
Date:   Thu Apr 22 15:32:54 2021 -0400

    Merge pull request #3 from ibarral18/ivan-testing

    ivan-testing

commit 2e3cbd12cc445d2ff2dc61036b56cfc78d900eb3
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Thu Apr 22 15:31:15 2021 -0400

    adding Matrix creation

commit 2b767a822a24b148309e7f81847082c29746fec3
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Thu Apr 15 20:30:08 2021 -0400

    fixing minor errors

commit f2b0dd0606bb372b11a996c9214645e991985900
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Thu Apr 15 01:54:57 2021 -0400

    adding one line var decl and matrix type check

commit 081b2dbce74abada58babdd2af4e05fcc10fdfbf
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Sat Apr 10 17:10:14 2021 +0100

    switching to when for matrix check

commit 179c0d7e35e95488ea7b7dcab5711b9ad296d835
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Thu Apr 8 02:11:50 2021 +0100
```

```
    Error checking edits

commit 12b7bccd611c7ecc1bf8cc1e431f647937934 86c
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Thu Apr 8 00:30:00 2021 +0100

    Minor edits for compiling

commit 395b7f59dce257af3506508b0a8ff7686615 36a2
Author: Rafail Khalilov <"khalilovrafail1@gmail.com">
Date:   Wed Apr 7 11:26:50 2021 -0400

    updated parser - no conflicts + matrix assign

commit 9b30e5aea17cbd42949541dba91b56784524aaaa
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Wed Apr 7 01:41:07 2021 +0100

    Added FMatrix to Parser

commit 83b23a11a995b30fc3011886973cce2172c5ab91
Author: Rafail Khalilov <"khalilovrafail1@gmail.com">
Date:   Tue Apr 6 20:14:49 2021 -0400

    Updated scanner + parser

commit bc7e8a8ca55f48cc407e87912c0d3faf7b0a98af
Merge: 07ea163 25bb62f
Author: Rafail Khalilov <"khalilovrafail1@gmail.com">
Date:   Tue Apr 6 19:53:14 2021 -0400

    Merge branch 'hello' of https://github.com/ibarral18/Red-Pandas into
hello

commit 07ea1630ac44273 99fd82f95471beec5db89a9fc
Author: Rafail Khalilov <"khalilovrafail1@gmail.com">
Date:   Tue Apr 6 19:51:22 2021 -0400

    updated parser

commit 25bb62fff3a5f103e5571574bd6aaecceef394cf
Author: Rafail Khalilov <khalilovrafail1@gmail.com>
Date:   Fri Apr 2 23:16:15 2021 -0400
```

```
    Update README.md

commit 5b7b9f51aa5e475117b11da207c73c4bc39f5f1a
Author: Rafail Khalilov <khalilovrafail1@gmail.com>
Date:   Fri Apr 2 23:15:41 2021 -0400

    Update README.md

commit 7c3117fe22b33a145abeb1662785d60c1bbb5241
Author: Rafail Khalilov <"khalilovrafail1@gmail.com">
Date:   Fri Apr 2 23:11:35 2021 -0400

    Matrix data types, no code-gen, semantic checking only

commit ceeb620c37eedcbffeb7213f105bd30bc976b826
Author: Rafail Khalilov <"khalilovrafail1@gmail.com">
Date:   Fri Apr 2 22:59:11 2021 -0400

    Matrix data type, no code-gen, only semantical checking

commit 6f9fd28da063b7a6d8974769c2389137b41524772
Merge: 1ecccd7 e3ff52c
Author: myric <18725865+myric@users.noreply.github.com>
Date:   Thu Apr 1 01:24:15 2021 +0100

    Merge pull request #2 from ibarral18/hello

    Hello

commit e3ff52ce4a1634da88ff01fad2a825e97a2b6ca0
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Thu Mar 25 00:10:50 2021 +0000

    Updates to align file names and extensions in Readme and Makefile

commit 679a05ab913f0b43e756e92bc0657a7d01f871991
Author: Rafail Khalilov <khalilovrafail1@gmail.com>
Date:   Tue Mar 23 17:25:11 2021 -0400

    Update README.md

commit a84538f1e152b238b74f62017e64b1f9757cf061
```

```
Merge: bf385c9 9f24de9
Author: aassal <assal.amina.h@gmail.com>
Date:   Tue Mar 23 14:24:19 2021 -0700

    Merge branch 'hello' of https://github.com/ibarral18/Red-Pandas into
hello

commit bf385c928522bc7cb3451de3ac7b0090f086c8ef
Author: aassal <assal.amina.h@gmail.com>
Date:   Tue Mar 23 14:23:43 2021 -0700

    changed hello world to rp

commit 9f24de9e3b3cd065fdf72be77b79b2c53f43c7f8
Author: myric <18725865+myric@users.noreply.github.com>
Date:   Tue Mar 23 21:17:14 2021 +0000

    Update README.md

commit 329d7ccaac193d072bf685e3108c461003ed383ea
Author: Myric Lehner <18725865+myric@users.noreply.github.com>
Date:   Tue Mar 23 21:14:29 2021 +0000

    removed extra microc tests

commit 92a2f1b7304e7381589cd13bf9b4f92a1f0de057
Author: aassal <assal.amina.h@gmail.com>
Date:   Tue Mar 23 12:18:04 2021 -0700

    changes file extension thing in redpandas.ml, Makefile, and test script

commit 09aaa45b3456cefa93391f2d868f5392baf295113
Merge: c440632 f90a468
Author: aassal <assal.amina.h@gmail.com>
Date:   Tue Mar 23 12:14:39 2021 -0700

    Merge branch 'hello' of https://github.com/ibarral18/Red-Pandas into
hello

commit c4406327b20de2df17d387ea99f8dcd4f04d761c
Author: aassal <assal.amina.h@gmail.com>
Date:   Tue Mar 23 12:13:43 2021 -0700
```

```
    changed test file extensions to rp

commit 1ecccd7b72a22c91ce079048c4e7976377065c61
Merge: 32ef3b2 f90a468
Author: Ivan Barral <51125955+ibarral18@users.noreply.github.com>
Date:   Mon Mar 22 18:06:24 2021 -0400

    Merge pull request #1 from ibarral18/hello

    Hello

commit f90a468269ab8bd06fdbc1ac342200e2ac38d75c
Merge: 00a10d9 32ef3b2
Author: Ivan Barral <51125955+ibarral18@users.noreply.github.com>
Date:   Mon Mar 22 18:06:13 2021 -0400

    Merge branch 'main' into hello

commit 00a10d9b70c0dae4b42804b39f1a457be7c655fb
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Sat Mar 20 21:52:44 2021 -0400

    fixing hello world

commit 6026e0979e81d8545f681aad735e749a72dbae35
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Sat Mar 20 21:48:29 2021 -0400

    adding string & printstr capabilities to rp

commit 5107699520d52edfbe202cfd63193b0d04b78d61
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Sat Mar 20 21:12:34 2021 -0400

    fixing hello world and readme

commit 189fca2cfad3058bf8577c551003190244ad2190
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Sat Mar 20 21:00:54 2021 -0400

    update readme

commit d1ff8bb39cead792b9567d8c459b31c307cd80cb
```

```
Author: Ivan Barral <51125955+ibarral18@users.noreply.github.com>
Date:   Sat Mar 20 20:54:05 2021 -0400

    Update README.md

commit 32ef3b2adcc7b1be047c67b176e054ce7ec8cd26
Author: Ivan Barral <51125955+ibarral18@users.noreply.github.com>
Date:   Sat Mar 20 20:53:16 2021 -0400

    Update README.md

commit c1045fa4364a7594591f197c92abfaacda0cdecb
Author: Ivan Barral <51125955+ibarral18@users.noreply.github.com>
Date:   Sat Mar 20 20:52:43 2021 -0400

    Update README.md

commit 82d29ee3c129ad43c6ae89d97386e58db4b66b21
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Sat Mar 20 20:50:39 2021 -0400

    getting rid of some junk in first commit

commit 21b2477676247a33f002b7eb678eca656cb6b6e6
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Sat Mar 20 20:43:16 2021 -0400

    fixing error in testing

commit f7d719177bc34f47101ca8a7cdfd7664f3872e31
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Sat Mar 20 20:36:27 2021 -0400

    adding microC stuff to our code for hello world

commit a298a78759f993e05541cdeca9843c6b196610cc
Author: Rafail Khalilov <khalilovrafail1@gmail.com>
Date:   Tue Mar 16 21:03:41 2021 -0400

    hello world

commit 9a817f3a0bbc7145eb74b7db137d62d8b75b9673
Author: Myric Lehner <myriclehner@Aluminium.cust.communityfibre.co.uk>
```

```
Date:   Thu Feb 25 02:03:59 2021 +0000

    Got parser working with Menhir

commit 5c2bc3388c5036150 30f990ed7f5787ce0d3253c
Author: Myric Lehner <myriclehner@Aluminium.cust.communityfibre.co.uk>
Date:   Thu Feb 25 00:56:38 2021 +0000

    minor edit

commit f3b2f0fc9a4838db953212c5e138223f6f57eda1
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Wed Feb 24 19:32:54 2021 -0500

    adding scanner and minor parser fixes

commit c4be47d2afdf5cec296579b100fa926b2eca6329
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Wed Feb 24 17:55:01 2021 -0500

    adding mircoC components to parser

commit 0ec3c15628c25af237936f296facd4616350a8ad
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Tue Feb 23 20:02:26 2021 -0500

    parser additions

commit a9b682708c1da550963bf10c4823b000b28d19ec
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Sun Feb 21 22:06:48 2021 -0500

    readme

commit 62d9bfa49f88804a45eff2ad983682c29272eeea
Author: ibarral18 <ivanbarral18@gmail.com>
Date:   Sun Feb 21 22:04:53 2021 -0500

    initial commit
```

8.3 Test Suite

```
_____________________________________________________________________
_____________
tests/test-add1.rp
_____________________________________________________________________
_____________

def int add(int x, int y)
{
  return x + y;
}

def int main()
{
  print( add(17, 25) );
  return 0;
}


_____________________________________________________________________
_____________
tests/test-arith1.rp
_____________________________________________________________________
_____________

def int main()
{
  print(39 + 3);
  return 0;
}


_____________________________________________________________________
_____________
tests/test-arith2.rp
_____________________________________________________________________
_____________

def int main()
{
```

```
  print(1 + 2 * 3 + 4);
  return 0;
}
```

```
_____________________________________________________________________
_____________
tests/test-arith3.rp
_____________________________________________________________________
_____________

def int foo(int a)
{
  return a;
}

def int main()
{
  int a;
  a = 42;
  a = a + 5;
  print(a);
  return 0;
}
```

```
_____________________________________________________________________
_____________
tests/test-fib.rp
_____________________________________________________________________
_____________

def int fib(int x)
{
  if (x < 2) return 1;
  return fib(x-1) + fib(x-2);
}

def int main()
{
  print(fib(0));
```

```
  print(fib(1));
  print(fib(2));
  print(fib(3));
  print(fib(4));
  print(fib(5));
  return 0;
}
```

---

tests/test-float1.rp

```
def int main()
{
  float a;
  a = 3.14159267;
  printf(a);
  return 0;
}
```

---

tests/test-float2.rp

```
def int main()
{
  float a;
  float b;
  float c;
  a = 3.14159267;
  b = -2.71828;
  c = a + b;
  printf(c);
  return 0;
}
```

tests/test-float3.rp

```
def void testfloat(float a, float b)
{
  printf(a + b);
  printf(a - b);
  printf(a * b);
  printf(a / b);
  printb(a == b);
  printb(a == a);
  printb(a != b);
  printb(a != a);
  printb(a > b);
  printb(a >= b);
  printb(a < b);
  printb(a <= b);
}

def int main()
{
  float c;
  float d;

  c = 42.0;
  d = 3.14159;

  testfloat(c, d);

  testfloat(d, d);

  return 0;
}
```

```
_________________
tests/test-for1.rp
_____________________________________________________________________________
_________________

def int main()
{
  int i;
  for (i = 0 ; i < 5 ; i = i + 1) {
    print(i);
  }
  print(42);
  return 0;
}


_____________________________________________________________________________
_________________
tests/test-for2.rp
_____________________________________________________________________________
_________________

def int main()
{
  int i;
  i = 0;
  for ( ; i < 5; ) {
    print(i);
    i = i + 1;
  }
  print(42);
  return 0;
}


_____________________________________________________________________________
_________________
tests/test-func1.rp
_____________________________________________________________________________
_________________
```

```
def int add(int a, int b)
{
  return a + b;
}

def int main()
{
  int a;
  a = add(39, 3);
  print(a);
  return 0;
}
```

---

tests/test-func2.rp

```
/* Bug noticed by Pin-Chin Huang */

def int fun(int x, int y)
{
  return 0;
}

def int main()
{
  int i;
  i = 1;

  fun(i = 2, i = i+1);

  print(i);
  return 0;
}
```

```
_________________
tests/test-func3.rp
_____________________________________________________________________________
_________________

def void printem(int a, int b, int c, int d)
{
  print(a);
  print(b);
  print(c);
  print(d);
}

def int main()
{
  printem(42,17,192,8);
  return 0;
}


_____________________________________________________________________________
_________________
tests/test-func4.rp
_____________________________________________________________________________
_________________

def int add(int a, int b)
{
  int c;
  c = a + b;
  return c;
}

def int main()
{
  int d;
  d = add(52, 10);
  print(d);
  return 0;
}
```

```
_____
tests/test-func5.rp
_____

def int foo(int a)
{
  return a;
}

def int main()
{
  return 0;
}

_____
tests/test-func6.rp
_____

def void foo() {}

def int bar(int a, bool b, int c) { return a + c; }

def int main()
{
  print(bar(17, false, 25));
  return 0;
}

_____
tests/test-func7.rp
_____
```

```
int a;

def void foo(int c)
{
  a = c + 42;
}

def int main()
{
  foo(73);
  print(a);
  return 0;
}


_____
_____
tests/test-func8.rp
_____
_____

def void foo(int a)
{
  print(a + 3);
}

def int main()
{
  foo(40);
  return 0;
}


_____
_____
tests/test-func9.rp
_____
_____

def void foo(int a)
{
```

```
  print(a + 3);
  return;
}

def int main()
{
  foo(40);
  return 0;
}
```

---

tests/test-gcd.rp

```
def int gcd(int a, int b) {
  while (a != b) {
    if (a > b) a = a - b;
    else b = b - a;
  }
  return a;
}

def int main()
{
  print(gcd(2,14));
  print(gcd(3,15));
  print(gcd(99,121));
  return 0;
}
```

---

tests/test-gcd2.rp

```
def int gcd(int a, int b) {
```

```
  while (a != b)
    if (a > b) a = a - b;
    else b = b - a;
  return a;
}

def int main()
{
  print(gcd(14,21));
  print(gcd(8,36));
  print(gcd(99,121));
  return 0;
}
```

---

tests/test-global1.rp

```
int a;
int b;

def void printa()
{
  print(a);
}

def void printbb()
{
  print(b);
}

def void incab()
{
  a = a + 1;
  b = b + 1;
}
```

```
def int main()
{
  a = 42;
  b = 21;
  printa();
  printbb();
  incab();
  printa();
  printbb();
  return 0;
}
```

---

tests/test-global2.rp

```
bool i;

def int main()
{
  int i; /* Should hide the global i */

  i = 42;
  print(i + i);
  return 0;
}
```

---

tests/test-global3.rp

```
int i;
bool b;
int j;
```

```
def int main()
{
  i = 42;
  j = 10;
  print(i + j);
  return 0;
}


_____________________________________________________________________
_________________
tests/test-hello.rp
_____________________________________________________________________
_________________

def int main()
{
  print(42);
  print(71);
  print(1);
  return 0;
}


_____________________________________________________________________
_________________
tests/test-if1.rp
_____________________________________________________________________
_________________

def int main()
{
  if (true) print(42);
  print(17);
  return 0;
}


_____________________________________________________________________
_________________
tests/test-if2.rp
_____________________________________________________________________
```

```
_________________

def int main()
{
  if (true) print(42); else print(8);
  print(17);
  return 0;
}

_____________________________________________________________________
_________________
tests/test-if3.rp
_____________________________________________________________________
_________________

def int main()
{
  if (false) print(42);
  print(17);
  return 0;
}

_____________________________________________________________________
_________________
tests/test-if4.rp
_____________________________________________________________________
_________________

def int main()
{
  if (false) print(42); else print(8);
  print(17);
  return 0;
}

_____________________________________________________________________
_________________
tests/test-if5.rp
_____________________________________________________________________
```

```
_________________

def int cond(bool b)
{
  int x;
  if (b)
    x = 42;
  else
    x = 17;
  return x;
}

def int main()
{
 print(cond(true));
 print(cond(false));
 return 0;
}
```

```
_____________________________________________________________________________
_________________
tests/test-if6.rp
_____________________________________________________________________________
_________________
```

```
def int cond(bool b)
{
  int x;
  x = 10;
  if (b)
    if (x == 10)
      x = 42;
  else
    x = 17;
  return x;
}

def int main()
{
```

```
 print(cond(true));
 print(cond(false));
 return 0;
}
```

```
_____________________________________________________________________
_____________
tests/test-local1.rp
_____________________________________________________________________
_____________
```

```
def void foo(bool i)
{
  int i; /* Should hide the formal i */

  i = 42;
  print(i + i);
}

def int main()
{
  foo(true);
  return 0;
}
```

```
_____________________________________________________________________
_____________
tests/test-local2.rp
_____________________________________________________________________
_____________
```

```
def int foo(int a, bool b)
{
  int c;
  bool d;

  c = a;

  return c + 10;
```

```
}

def int main() {
 print(foo(37, false));
 return 0;
}


_____
tests/test-matrix-access.rp
_____

def int main()
{
    matrix int [3][3] x;

    x = [[1,2,3],
         [4,5,6],
         [7,8,9]];

    print(x[0][1]);


  return 0;
}
_____
tests/test-matrix-add1.rp
_____

def matrix int [3][3] add(matrix int [3][3] x, matrix int [3][3] y)
{
  return x + y;
}

def int main()
{
```

```
    int i;
    int j;
    matrix int [3][3] x;
    matrix int [3][3] y;
    matrix int [3][3] sum;


     x = [[1,2,3],
         [4,5,6],
         [7,8,9]];

    y = [[9,8,7],
         [6,5,4],
         [3,2,1]];

    sum = add(x,y);

    for(j = 0; j < sum.row; j = j + 1 ){
             for(i = 0; i < sum.col; i = i + 1){
                print(sum[j][i]);
             }
      printStr("");
       }

  return 0;
}
```

---

tests/test-matrix-mult1.rp

```
def matrix int [3][3] mult(matrix int [3][3] x, matrix int [3][3] y)
{
  return x * y;
}

def int main()
{
```

```
    int i;
    int j;
    matrix int [3][3] x;
    matrix int [3][3] y;
    matrix int [3][3] prod;



     x = [[1,2,3],
         [4,5,6],
         [7,8,9]];

    y = [[9,8,7],
         [6,5,4],
         [3,2,1]];

    prod = mult(x,y);

    for(j = 0; j < prod.row; j = j + 1 ){
             for(i = 0; i < prod.col; i = i + 1){
                print(prod[j][i]);
             }
        printStr("");
       }


  return 0;
}
```

---

tests/test-matrix-mult3.rp

```
def matrix int [2][2] mult(matrix int [2][3] x, matrix int [3][2] y)
{
  return x * y;
}
```

```
def int main()
{
    int i;
    int j;
    matrix int [2][3] x;
    matrix int [3][2] y;
    matrix int [2][2] prod;



     x = [[1,2,3],
         [4,5,6]];

    y = [[9,8],
         [6,5],
         [3,2]];

    prod = mult(x,y);

    for(j = 0; j < prod.row; j = j + 1 ){
             for(i = 0; i < prod.col; i = i + 1){
                print(prod[j][i]);
             }
      printStr("");
       }


  return 0;
}
```

---

tests/test-matrix-sub1.rp

```
def matrix int [3][3] sub(matrix int [3][3] x, matrix int [3][3] y)
{
  return x - y;
}
```

```
def int main()
{
    int i;
    int j;
    matrix int [3][3] x;
    matrix int [3][3] y;
    matrix int [3][3] diff;



     x = [[1,2,3],
         [4,5,6],
         [7,8,9]];

    y = [[9,8,7],
         [6,5,4],
         [3,2,1]];

    diff = sub(x,y);


  for(j = 0; j < diff.row; j = j + 1 ){
           for(i = 0; i < diff.col; i = i + 1){
                print(diff[j][i]);
           }
       printStr("");
     }
  return 0;
}
```

---

tests/test-matrix-transpose1.rp

```
def int main()
{
    int i;
```

```
    int j;
    matrix int [3][3] x;
    matrix int [3][3] y;

    x = [[1,2,3],
         [4,5,6],
         [7,8,9]];

    y = x.T;

    for(j = 0; j < y.row; j = j + 1 ){
             for(i = 0; i < y.col; i = i + 1){
                print(y[j][i]);
             }
      printStr("");
        }


  return 0;
}
```

---

tests/test-matrix-transpose2.rp

```
def int main()
{
    int i;
    int j;
    matrix int [3][4] x;
    matrix int [4][3] y;

    x = [[1,2,3,4],
         [5,6,7,8],
         [9,10,11,12]];

    y = x.T;
```

```
    for(j = 0; j < y.row; j = j + 1 ){
             for(i = 0; i < y.col; i = i + 1){
                  print(y[j][i]);
             }
       printStr("");
        }


  return 0;
}
```

---

tests/test-ops1.rp

```
def int main()
{
  print(1 + 2);
  print(1 - 2);
  print(1 * 2);
  print(100 / 2);
  print(99);
  printb(1 == 2);
  printb(1 == 1);
  print(99);
  printb(1 != 2);
  printb(1 != 1);
  print(99);
  printb(1 < 2);
  printb(2 < 1);
  print(99);
  printb(1 <= 2);
  printb(1 <= 1);
  printb(2 <= 1);
  print(99);
  printb(1 > 2);
  printb(2 > 1);
  print(99);
```

```
  printb(1 >= 2);
  printb(1 >= 1);
  printb(2 >= 1);
  return 0;
}
```

---

tests/test-ops2.rp

```
def int main()
{
  printb(true);
  printb(false);
  printb(true && true);
  printb(true && false);
  printb(false && true);
  printb(false && false);
  printb(true || true);
  printb(true || false);
  printb(false || true);
  printb(false || false);
  printb(!false);
  printb(!true);
  print(-10);
  print(--42);
}
```

---

tests/test-printbig.rp

```
/*
 * Test for linking external C functions to LLVM-generated code
 *
```

```
 * printbig is defined as an external function, much like printf
 * The C compiler generates printbig.o
 * The LLVM compiler, llc, translates the .ll to an assembly .s file
 * The C compiler assembles the .s file and links the .o file to
generate
 * an executable
 */

def int main()
{
  printbig(72); /* H */
  printbig(69); /* E */
  printbig(76); /* L */
  printbig(76); /* L */
  printbig(79); /* O */
  printbig(32); /*   */
  printbig(87); /* W */
  printbig(79); /* O */
  printbig(82); /* R */
  printbig(76); /* L */
  printbig(68); /* D */
  return 0;
}
```

---

tests/test-var1.rp

```
def int main()
{
  int a;
  a = 42;
  print(a);
  return 0;
}
```

```
_________________
tests/test-var2.rp
_____________________________________________________________________________
_________________

int a;

def void foo(int c)
{
  a = c + 42;
}

def int main()
{
  foo(73);
  print(a);
  return 0;
}


_____________________________________________________________________________
_________________
tests/test-while1.rp
_____________________________________________________________________________
_________________

def int main()
{
  int i;
  i = 5;
  while (i > 0) {
    print(i);
    i = i - 1;
  }
  print(42);
  return 0;
}


_____________________________________________________________________________
_________________
```

tests/test-while2.rp

```
def int foo(int a)
{
  int j;
  j = 0;
  while (a > 0) {
    j = j + 2;
    a = a - 1;
  }
  return j;
}

def int main()
{
  print(foo(7));
  return 0;
}
```

tests/fail-assign1.rp

```
def int main()
{
  int i;
  bool b;

  i = 42;
  i = 10;
  b = true;
  b = false;
  i = false; /* Fail: assigning a bool to an integer */
}
```

```
_____
tests/fail-assign2.rp
_____

def int main()
{
  int i;
  bool b;

  b = 48; /* Fail: assigning an integer to a bool */
}


_____
tests/fail-assign3.rp
_____

def void myvoid()
{
  return;
}

def int main()
{
  int i;

  i = myvoid(); /* Fail: assigning a void to an integer */
}


_____
tests/fail-dead1.rp
_____

def int main()
```

```
{
  int i;

  i = 15;
  return i;
  i = 32; /* Error: code after a return */
}
```

---

tests/fail-dead2.rp

```
def int main()
{
  int i;

  {
    i = 15;
    return i;
  }
  i = 32; /* Error: code after a return */
}
```

---

tests/fail-expr1.rp

```
int a;
bool b;

def void foo(int c, bool d)
{
  int dd;
  bool e;
  a + c;
```

```
  c - a;
  a * 3;
  c / 2;
  d + a; /* Error: bool + int */
}

def int main()
{
  return 0;
}
```

```
_____________________________________________________________________
_____________
tests/fail-expr2.rp
_____________________________________________________________________
_____________

int a;
bool b;

def void foo(int c, bool d)
{
  int d;
  bool e;
  b + a; /* Error: bool + int */
}

def int main()
{
  return 0;
}
```

```
_____________________________________________________________________
_____________
tests/fail-expr3.rp
_____________________________________________________________________
_____________

int a;
```

```
float b;

def void foo(int c, float d)
{
  int d;
  float e;
  b + a; /* Error: float + int */
}

def int main()
{
  return 0;
}
```

```
_____
tests/fail-float1.rp
_____
```

```
def int main()
{
  -3.5 && 1; /* Float with AND? */
  return 0;
}
```

```
_____
tests/fail-float2.rp
_____
```

```
def int main()
{
  -3.5 && 2.5; /* Float with AND? */
  return 0;
}
```

```
_____
```

```
_________________
tests/fail-for1.rp
_____________________________________________________________________________
_________________

def int main()
{
  int i;
  for ( ; true ; ) {} /* OK: Forever */

  for (i = 0 ; i < 10 ; i = i + 1) {
    if (i == 3) return 42;
  }

  for (j = 0; i < 10 ; i = i + 1) {} /* j undefined */

  return 0;
}


_____________________________________________________________________________
_________________
tests/fail-for2.rp
_____________________________________________________________________________
_________________

def int main()
{
  int i;

  for (i = 0; j < 10 ; i = i + 1) {} /* j undefined */

  return 0;
}


_____________________________________________________________________________
_________________
tests/fail-for3.rp
_____________________________________________________________________________
_________________
```

```
def int main()
{
  int i;

  for (i = 0; i ; i = i + 1) {} /* i is an integer, not Boolean */

  return 0;
}
```

---

tests/fail-for4.rp

```
def int main()
{
  int i;

  for (i = 0; i < 10 ; i = j + 1) {} /* j undefined */

  return 0;
}
```

---

tests/fail-for5.rp

```
def int main()
{
  int i;

  for (i = 0; i < 10 ; i = i + 1) {
    foo(); /* Error: no function foo */
  }
```

```
  return 0;
}


_____
tests/fail-func1.rp
_____

def int foo() {}

def int bar() {}

def int baz() {}

def void bar() {} /* Error: duplicate function bar */

def int main()
{
  return 0;
}


_____
tests/fail-func2.rp
_____

def int foo(int a, bool b, int c) { }

def void bar(int a, bool b, int a) {} /* Error: duplicate formal a in
bar */

def int main()
{
  return 0;
}


_____
```

```
_________________
tests/fail-func3.rp
_____________________________________________________________________________
_________________

def int foo(int a, bool b, int c) { }

def void bar(int a, void b, int c) {} /* Error: illegal void formal b
*/

def int main()
{
  return 0;
}


_____________________________________________________________________________
_________________
tests/fail-func4.rp
_____________________________________________________________________________
_________________

def int foo() {}

def void bar() {}

def int print() {} /* Should not be able to define print */

def void baz() {}

def int main()
{
  return 0;
}


_____________________________________________________________________________
_________________
tests/fail-func5.rp
_____________________________________________________________________________
_________________
```

```
def int foo() {}

def int bar() {
  int a;
  void b; /* Error: illegal void local b */
  bool c;

  return 0;
}

def int main()
{
  return 0;
}
```

```
tests/fail-func6.rp
```

```
def void foo(int a, bool b)
{
}

def int main()
{
  foo(42, true);
  foo(42); /* Wrong number of arguments */
}
```

```
tests/fail-func7.rp
```

```
def void foo(int a, bool b)
```

```
{
}

def int main()
{
  foo(42, true);
  foo(42, true, false); /* Wrong number of arguments */
}
```

---

tests/fail-func8.rp

```
def void foo(int a, bool b)
{
}

def void bar()
{
}

def int main()
{
  foo(42, true);
  foo(42, bar()); /* int and void, not int and bool */
}
```

---

tests/fail-func9.rp

```
def void foo(int a, bool b)
{
}
```

```
def int main()
{
  foo(42, true);
  foo(42, 42); /* Fail: int, not bool */
}
```

---

tests/fail-global1.rp

```
int c;
bool b;
void a; /* global variables should not be void */


def int main()
{
  return 0;
}
```

---

tests/fail-global2.rp

```
int b;
bool c;
int a;
int b; /* Duplicate global variable */

def int main()
{
  return 0;
}
```

```
_________________
tests/fail-if1.rp
_____________________________________________________________________________
_________________

def int main()
{
  if (true) {}
  if (false) {} else {}
  if (42) {} /* Error: non-bool predicate */
}


_____________________________________________________________________________
_________________
tests/fail-if2.rp
_____________________________________________________________________________
_________________

def int main()
{
  if (true) {
    foo; /* Error: undeclared variable */
  }
}


_____________________________________________________________________________
_________________
tests/fail-if3.rp
_____________________________________________________________________________
_________________

def int main()
{
  if (true) {
    42;
  } else {
    bar; /* Error: undeclared variable */
  }
}
```

```
_____________________________________________________________________
_________________
tests/fail-matrix-access.rp
_____________________________________________________________________
_________________

def int main()
{
    int i;
    int j;
    matrix int [3][3] x;
    matrix int [3][3] y;

    x = [[1,2,3],
         [4,5,6],
         [7,8,9]];

    print(x[2.0][0]);


  return 0;
}
_____________________________________________________________________
_________________
tests/fail-matrix-add1.rp
_____________________________________________________________________
_________________

def matrix int [3][3] add(matrix int [3][3] x, matrix int [3][3] y)
{
  return x + y;
}

def int main()
{
    int i;
    int j;
    matrix int [3][3] x;
```

```
    matrix int [3][3] y;
    matrix int [3][3] sum;


      x = [[1.0,2.0,3.0],
          [4.0,5.0,6.0],
          [7.0,8.0,9.0]];

    y = [[9,8,7],
         [6,5,4],
         [3,2,1]];

    sum = add(x,y);

    for(j = 0; j < sum.row; j = j + 1 ){
           for(i = 0; i < sum.col; i = i + 1){
                print(sum[j][i]);
           }
       printStr("");
     }


  return 0;
}
```

---

tests/fail-matrix-add2.rp

```
def matrix int [3][3] add(matrix int [3][3] x, matrix float [3][3] y)
{
  return x + y;
}

def int main()
{
    int i;
    int j;
```

```
    matrix int [3][3] x;
    matrix int [3][3] y;
    matrix int [3][3] sum;


     x = [[1,2,3],
         [4,5,6],
         [7,8,9]];

    y = [[9.0,8.0,7.0],
         [6.0,5.0,4.0],
         [3.0,2.0,1.0]];


    sum = add(x, y);

    for(j = 0; j < sum.row; j = j + 1 ){
           for(i = 0; i < sum.col; i = i + 1){
                print(sum[j][i]);
           }
       printStr("");
     }

  return 0;
}
```

---

tests/fail-matrix-add3.rp

```
def matrix int [3][3] add(matrix int [3][3] x, matrix int [2][2] y)
{
  return x + y;
}

def int main()
{
    int i;
```

```
    int j;
    matrix int [3][3] x;
    matrix int [2][2] y;
    matrix int [3][3] sum;


      x = [[1,2,3],
          [4,5,6],
          [7,8,9]];

    y = [[9,8],
         [6,5]];

    sum = add(x,y);

    for(j = 0; j < sum.row; j = j + 1 ){
           for(i = 0; i < sum.col; i = i + 1){
                print(sum[j][i]);
           }
        printStr("");
     }

  return 0;
}
```

---

tests/fail-matrix-add4.rp

```
def matrix int [3][3] add(matrix int [3][3] x, matrix int [3][3] y,
matrix int [3][3] z)
{
  return x + y + z;
}

def int main()
{
    int i;
```

```
    int j;
    matrix int [3][3] x;
    matrix int [3][3] y;
    matrix int [3][3] z;
    matrix int [3][3] sum;



      x = [[1,2,3],
          [4,5,6],
          [7,8,9]];

    y = [[9,8,7],
         [6,5,4],
         [3,2,1]];

    z = [[1,1,1],
         [1,1,1],
         [1,1,1]];

    sum = add(x,y,z);

    for(j = 0; j < sum.row; j = j + 1 ){
           for(i = 0; i < sum.col; i = i + 1){
                print(sum[j][i]);
           }
       printStr("");
     }

  return 0;
}
```

---

tests/fail-matrix-arith.rp

```
def int main()
{
```

```
    int i;
    int j;
    matrix int [3][3] x;
    matrix int [3][3] y;
    matrix int [3][3] total;


     x = [[1,2,3],
         [4,5,6],
         [7,8,9]];

    y = [[9,8,7],
         [6,5,4],
         [3,2,1]];

    total = x + y * x;

    for(j = 0; j < total.row; j = j + 1 ){
           for(i = 0; i < total.col; i = i + 1){
                print(total[j][i]);
           }
       printStr("");
     }

  return 0;
}
```

---

tests/fail-matrix-mult1.rp

```
def matrix int [3][3] mult(matrix int [3][3] x, matrix int [3][3] y)
{
  return x * y;
}

def int main()
{
```

```
    int i;
    int j;
    matrix int [3][3] x;
    matrix int [3][3] y;
    matrix int [3][3] prod;


     x = [[1.0,2.0,3.0],
         [4.0,5.0,6.0],
         [7.0,8.0,9.0]];

    y = [[9,8,7],
         [6,5,4],
         [3,2,1]];

    prod = mult(x,y);

    for(j = 0; j < prod.row; j = j + 1 ){
           for(i = 0; i < prod.col; i = i + 1){
                print(prod[j][i]);
           }
       printStr("");
     }


  return 0;
}
```

---

tests/fail-matrix-mult2.rp

```
def matrix int [3][3] mult(matrix int [3][3] x, matrix float [3][3]
y)
{
  return x * y;
}
```

```
def int main()
{
    int i;
    int j;
    matrix int [3][3] x;
    matrix int [3][3] y;
    matrix int [3][3] prod;

     x = [[1,2,3],
         [4,5,6],
         [7,8,9]];

    y = [[9.0,8.0,7.0],
         [6.0,5.0,4.0],
         [3.0,2.0,1.0]];

    prod = mult(x,y);


  for(j = 0; j < prod.row; j = j + 1 ){
          for(i = 0; i < prod.col; i = i + 1){
                print(prod[j][i]);
          }
       printStr("");
     }


  return 0;
}
```

---

tests/fail-matrix-mult3.rp

```
def matrix int [3][3] mult(matrix int [3][3] x, matrix int [2][2] y)
{
  return x * y;
}
```

```
def int main()
{
    int i;
    int j;
    matrix int [3][3] x;
    matrix int [2][2] y;
    matrix int [3][3] prod;


     x = [[1,2,3],
         [4,5,6],
         [7,8,9]];

    y = [[9,8],
         [6,5]];

    prod = mult(x,y);

    for(j = 0; j < prod.row; j = j + 1 ){
           for(i = 0; i < prod.col; i = i + 1){
                print(prod[j][i]);
           }
       printStr("");
     }

  return 0;
}
```

---

tests/fail-matrix-mult4.rp

```
def matrix int [3][3] mult(matrix int [3][3] x, matrix int [3][3] y,
matrix int [3][3] z)
{
  return x * y * z;
}
```

```
def int main()
{
    int i;
    int j;
    matrix int [3][3] x;
    matrix int [3][3] y;
    matrix int [3][3] z;
    matrix int [3][3] prod;



     x = [[1,2,3],
         [4,5,6],
         [7,8,9]];

    y = [[9,8,7],
         [6,5,4],
         [3,2,1]];

    z = [[2,2,2],
         [2,2,2],
         [2,2,2]];

    prod = mult(x,y,z);

    for(j = 0; j < prod.row; j = j + 1 ){
           for(i = 0; i < prod.col; i = i + 1){
                print(prod[j][i]);
           }
       printStr("");
     }


  return 0;
}
```

---

tests/fail-matrix-sub1.rp

```
_____________________________________________________________________
_________________

def matrix int [3][3] sub(matrix int [3][3] x, matrix int[3][3] y)
{
  return x - y;
}

def int main()
{
    int i;
    int j;
    matrix int [3][3] x;
    matrix int [3][3] y;
    matrix int [3][3] diff;

     x = [[1.0,2.0,3.0],
         [4.0,5.0,6.0],
         [7.0,8.0,9.0]];

    y = [[9,8,7],
         [6,5,4],
         [3,2,1]];

    diff = sub(x,y);


  for(j = 0; j < diff.row; j = j + 1 ){
          for(i = 0; i < diff.col; i = i + 1){
               print(diff[j][i]);
          }
       printStr("");
     }

  return 0;
}
_____________________________________________________________________
_________________
tests/fail-matrix-sub2.rp
```

```
_____
_____

def matrix int [3][3] sub(matrix int [3][3] x, matrix float [3][3] y)
{
  return x - y;
}

def int main()
{
    int i;
    int j;
    matrix int [3][3] x;
    matrix int [3][3] y;
    matrix int [3][3] diff;

     x = [[1,2,3],
         [4,5,6],
         [7,8,9]];

    y = [[9.0,8.0,7.0],
         [6.0,5.0,4.0],
         [3.0,2.0,1.0]];

    diff = sub(x,y);


  for(j = 0; j < diff.row; j = j + 1 ){
          for(i = 0; i < diff.col; i = i + 1){
               print(diff[j][i]);
          }
       printStr("");
     }


  return 0;
}
_____
_____
```

tests/fail-matrix-sub3.rp

```
_____________________________________________________________________________
_________________

def matrix int [3][3] sub(matrix int [3][3] x, matrix int [2][2] y)
{
  return x - y;
}

def int main()
{
    int i;
    int j;
    matrix int [3][3] x;
    matrix int [2][2] y;
    matrix int [3][3] diff;


     x = [[1,2,3],
         [4,5,6],
         [7,8,9]];

    y = [[9,8],
         [6,5]];

    diff = sub(x,y);

    for(j = 0; j < diff.row; j = j + 1 ){
           for(i = 0; i < diff.col; i = i + 1){
                print(diff[j][i]);
           }
        printStr("");
     }

  return 0;
}
_____________________________________________________________________________
_________________
```

tests/fail-matrix-sub4.rp

```
_____
_____

def matrix int [3][3] sub(matrix int [3][3] x, matrix int [3][3] y,
matrix int [3][3] z)
{
  return x - y - z;
}

def int main()
{
    int i;
    int j;
    matrix int [3][3] x;
    matrix int [3][3] y;
    matrix int [3][3] z;
    matrix int [3][3] diff;


     x = [[10,10,10],
         [10,10,10],
         [10,10,10]];

    y = [[9,8,7],
         [6,5,4],
         [3,2,1]];

    z = [[1,1,1],
         [1,1,1],
         [1,1,1]];

    diff = sub(x, y, z);

     for(j = 0; j < diff.row; j = j + 1 ){
           for(i = 0; i < diff.col; i = i + 1){
                print(diff[j][i]);
           }
       printStr("");
     }
```

```
  return 0;
}
```

---

tests/fail-matrix-transpose.rp

```
def int main()
{
    int i;
    int j;
    matrix int [3][4] x;
    matrix int [3][4] y;

    x = [[1,2,3],
         [4,5,6],
         [7,8,9]];

    y = x.T;

    for(j = 0; j < y.row; j = j + 1 ){
           for(i = 0; i < y.col; i = i + 1){
                print(y[j][i]);
           }
        printStr("");
     }


  return 0;
}
```

---

tests/fail-nomain.rp

```
_________________
tests/fail-print.rp
_____________________________________________________________________________
_________________

/* Should be illegal to redefine */
def void print() {}


_____________________________________________________________________________
_________________
tests/fail-printb.rp
_____________________________________________________________________________
_________________

/* Should be illegal to redefine */
def void printb() {}


_____________________________________________________________________________
_________________
tests/fail-printbig.rp
_____________________________________________________________________________
_________________

/* Should be illegal to redefine */
def void printbig() {}


_____________________________________________________________________________
_________________
tests/fail-return1.rp
_____________________________________________________________________________
_________________

def int main()
{
  return true; /* Should return int */
}


_____________________________________________________________________________
_________________
```

```
tests/fail-return2.rp
_____________________________________________________________________
_________________

def void foo()
{
  if (true) return 42; /* Should return void */
  else return;
}

def int main()
{
  return 42;
}


_____________________________________________________________________
_________________
tests/fail-string.rp
_____________________________________________________________________
_________________

def void main() {

    String test;

    test = "some words";

    print(test);
}
_____________________________________________________________________
_________________
tests/fail-while1.rp
_____________________________________________________________________
_________________

def int main()
{
  int i;
```

```
  while (true) {
    i = i + 1;
  }

  while (42) { /* Should be boolean */
    i = i + 1;
  }

}
```

---

tests/fail-while2.rp

```
def int main()
{
  int i;

  while (true) {
    i = i + 1;
  }

  while (true) {
    foo(); /* foo undefined */
  }

}
```