CMAT Final Report
COMS 4115

Language Guru: Michael Berkowitz (meb2235)
Project Manager: Frank Cabada (fc2452)
System Architect: Marissa Ojeda (mgo2111)
Tester: Daniel Rojas (dhr2119)

Contents

1. Introduction **7**
1.1 Motivation 7
1.2 Background 7

2. Tutorial **8**
2.1 Setting Up the Environment 8
2.2 Building and Using the Compiler 8
2.2.1 Building 8
2.2.2 Usage 9
2.3 The Basics 9
2.3.1 Primitives 9
2.3.2 Global Variables 10
2.3.3 Include Statement 10
2.3.4 Operators 10
2.3.5 Vectors and Matrices 11
2.3.6 Control Flow 11
2.3.7 Defining Functions 13

3. Language Reference Manual **14**
3.1 Types 14
3.1.1 Primitive Types 14
3.1.2 Type Casting 15
3.2 Lexical Conventions 15
3.2.1 Identifiers 15
3.2.2 Keywords 16
3.2.3 Operators 16
3.2.4 Precedence 17
3.2.5 Comments 18
3.3 Syntax Notation 18
3.3.1 Expressions 18
3.3.2 Declaration 19
3.3.3 Initialization 20
3.3.4 Statements 20
3.4 Standard Library Functions 22
3.4.1 Math 22
3.4.2 Vectors 22
3.4.3 Matrix 23

3.4.4 I/O 23
3.5 Semantics 24

4. Project Plan 25
4.1 Project Overview 25
4.2 Project Timeline 25
4.3 Challenges and Changes 27
4.4 Roles and Responsibilities 28
4.4.1 Roles 28
4.4.2 Team responsibilities 29
4.5 Development Environment 29
4.6 Version Control Statistics 30
4.6.1 Punch Card 30
4.6.2 Code Frequency 30
4.7 Master Branch Commit History 31

5. Translator Architecture 37
5.1 Diagram 37
5.2 Compiler 37
5.3 Prep 38
5.4 Scanner 38
5.5 Parser 38
5.6 Abstract Syntax Tree (AST) 39
5.7 Semantic Analyzer 39
5.8 Syntactically-Checked Abstract Syntax Tree (SAST) 40
5.9 Code Generator 40

6. Test Plan and Scripts 41
6.1 Travis CI 41
6.2 Scripts 41
6.2.1 Compiler 41
6.2.2 Parser 41
6.2.3 Scanner 42

7. Conclusions 43
7.1 Mike 43
7.2 Frank 43
7.3 Daniel 43
7.4 Marissa 44

A. Full Code Listing 45
A.1 plt/hello_world 45
A.1.1 plt/hello_world/scripts/build.sh 45
A.1.2 plt/hello_world/scripts/clean.sh 45
A.1.3 plt/hello_world/scripts/test.sh 45

A.1.4 plt/hello_world/hello_world.cmat 45
A.1.5 plt/hello_world/hello_world.out 46
A.2 plt/scripts 46
A.2.1 plt/scripts/build.sh 46
A.2.2 plt/scripts/clean.sh 46
A.2.3 plt/scripts/test.sh 46
A.3 plt/src 47
A.3.1 plt/src/ast.ml 47
A.3.2 plt/src/cmat.ml 48
A.3.3 plt/src/codegen.ml 49
A.3.4 plt/src/exceptions.ml 64
A.3.5 plt/src/parser.mly 66
A.3.6 plt/src/prep.ml 69
A.3.7 plt/src/sast.ml 70
A.3.8 plt/src/scanner.mll 71
A.3.9 plt/src/semant.ml 72
A.3.10 plt/src/utils.ml 81
A.4 plt/test 81
A.4.1 plt/test/compiler/fail 86
A.4.1.1 plt/test/compiler/fail/_dupe_global_local.test 86
A.4.1.2 plt/test/compiler/fail/_dupe_global_local.out 86
A.4.1.3 plt/test/compiler/fail/_duplicate_global.test 86
A.4.1.4 plt/test/compiler/fail/_duplicate_global.out 86
A.4.1.5 plt/test/compiler/fail/_function_nonexistent.test 86
A.4.1.6 plt/test/compiler/fail/_function_nonexistent.out 87
A.4.1.7 plt/test/compiler/fail/_illegal_float_equality.test 87
A.4.1.8 plt/test/compiler/fail/_illegal_float_equality.out 87
A.4.1.9 plt/test/compiler/fail/_incorrect_function_args.test 87
A.4.1.10 plt/test/compiler/fail/_incorrect_function_args.out 87
A.4.1.11 plt/test/compiler/fail/_matrix_equality.test 87
A.4.1.12 plt/test/compiler/fail/_matrix_equality.out 87
A.4.1.13 plt/test/compiler/fail/_matrix_mismatch_add.test 87
A.4.1.14 plt/test/compiler/fail/_matrix_mismatch_add.out 88
A.4.1.15 plt/test/compiler/fail/_no_return.test 88
A.4.1.16 plt/test/compiler/fail/_no_return.out 88
A.4.1.17 plt/test/compiler/fail/_return_mismatch.test 88
A.4.1.18 plt/test/compiler/fail/_return_mismatch.out 88
A.4.1.19 plt/test/compiler/fail/_void_formal.test 88
A.4.1.20 plt/test/compiler/fail/_void_formal.out 88
A.4.1.21 plt/test/compiler/fail/_void_global.test 88
A.4.1.22 plt/test/compiler/fail/_void_global.out 89
A.4.1.23 plt/test/compiler/fail/_void_local.test 89

A.4.1.24 plt/test/compiler/fail/_void_local.out 89
A.4.1.25 plt/test/compiler/fail/_void_return.test 89
A.4.1.26 plt/test/compiler/fail/_void_return.out 89
A.4.2 plt/test/compiler/pass 89
A.4.2.1 plt/test/compiler/pass/_arithmetic_binops.test 89
A.4.2.2 plt/test/compiler/pass/_arithmetic_binops.out 90
A.4.2.3 plt/test/compiler/pass/_assign.test 90
A.4.2.4 plt/test/compiler/pass/_assign.out 91
A.4.2.5 plt/test/compiler/pass/_control_flow.test 91
A.4.2.6 plt/test/compiler/pass/_control_flow.out 91
A.4.2.7 plt/test/compiler/pass/_func_call.test 91
A.4.2.8 plt/test/compiler/pass/_func_call.out 91
A.4.2.9 plt/test/compiler/pass/_inc.cmat 92
A.4.2.10 plt/test/compiler/pass/_include.test 92
A.4.2.11 plt/test/compiler/pass/_include.out 92
A.4.2.12 plt/test/compiler/pass/_matrices.test 92
A.4.2.13 plt/test/compiler/pass/_matrices.out 93
A.4.2.14 plt/test/compiler/pass/_mat_vec.test 93
A.4.2.15 plt/test/compiler/pass/_mat_vec.out 95
A.4.2.16 plt/test/compiler/pass/_return.test 96
A.4.2.17 plt/test/compiler/pass/_return.out 96
A.4.2.18 plt/test/compiler/pass/_vectors.test 96
A.4.2.19 plt/test/compiler/pass/_vectors.out 97
A.4.3 plt/test/compiler/scripts/build.sh 97
A.4.4 plt/test/compiler/scripts/clean.sh 97
A.4.5 plt/test/compiler/scripts/test.sh 97
A.4.6 plt/test/parser/fail 103
A.4.6.1 plt/test/parser/fail/_illegal_binop.test 103
A.4.6.2 plt/test/parser/fail/_illegal_binop.out 103
A.4.6.3 plt/test/parser/fail/_internal_fdecl.test 103
A.4.6.4 plt/test/parser/fail/_internal_fdecl.out 103
A.4.6.5 plt/test/parser/fail/_malformed_fdecl.test 104
A.4.6.6 plt/test/parser/fail/_malformed_fdecl.out 104
A.4.6.7 plt/test/parser/fail/_malformed_matrix_decl.test 104
A.4.6.8 plt/test/parser/fail/_malformed_matrix_decl.out 104
A.4.6.9 plt/test/parser/fail/_postfix_unop.test 104
A.4.6.10 plt/test/parser/fail/_postfix_unop.out 104
A.4.6.11 plt/test/parser/fail/_vdecl_without_semi.test 104
A.4.6.12 plt/test/parser/fail/_vdecl_without_semi.out 104
A.4.7 plt/test/parser/pass 104
A.4.7.1 plt/test/parser/pass/_base_parser.test 104
A.4.7.2 plt/test/parser/pass/_base_parser.out 104

A.4.7.3 plt/test/parser/pass/_expr.test 105
A.4.7.4 plt/test/parser/pass/_expr.out 105
A.4.7.5 plt/test/parser/pass/_formal_opts.test 108
A.4.7.6 plt/test/parser/pass/_format_opts.out 108
A.4.7.7 plt/test/parser/pass/_main_with_assign.test 109
A.4.7.8 plt/test/parser/pass/_main_with_assign.out 109
A.4.7.9 plt/test/parser/pass/_main_with_fdecl.test 110
A.4.7.10 plt/test/parser/pass/_main_with_fdecl.out 110
A.4.7.11 plt/test/parser/pass/_main_with_return.test 110
A.4.7.12 plt/test/parser/pass/_main_with_return.out 110
A.4.7.13 plt/test/parser/pass/_primitive_decls.test 111
A.4.7.14 plt/test/parser/pass/_primitive_decls.out 111
A.4.7.15 plt/test/parser/pass/_stmts.test 112
A.4.7.16 plt/test/parser/pass/_stmts.out 112
A.4.8 plt/test/parser/scripts/clean.sh 114
A.4.9 plt/test/parser/scripts/test.sh 114
A.4.10 plt/test/scanner/fail 114
A.4.10.1 plt/test/scanner/fail/_illegal_carrot.test 118
A.4.10.2 plt/test/scanner/fail/_illegal_carrot.out 118
A.4.10.3 plt/test/scanner/fail/_illegal_dollar.test 118
A.4.10.4 plt/test/scanner/fail/_illegal_dollar.out 118
A.4.10.5 plt/test/scanner/fail/_illegal_percent.test 118
A.4.10.6 plt/test/scanner/fail/_illegal_percent.out 118
A.4.10.7 plt/test/scanner/fail/_illegal_period.test 118
A.4.10.8 plt/test/scanner/fail/_illegal_period.out 118
A.4.10.9 plt/test/scanner/fail/_illegal_pound.test 118
A.4.10.10 plt/test/scanner/fail/_illegal_pound.out 118
A.4.10.11 plt/test/scanner/fail/_illegal_tilde.test 118
A.4.10.12 plt/test/scanner/fail/_illegal_tilde.out 119
A.4.11 plt/test/scanner/pass 119
A.4.11.1 plt/test/scanner/pass/_arithmetic.test 119
A.4.11.2 plt/test/scanner/pass/_arithmetic.out 119
A.4.11.3 plt/test/scanner/pass/_assignment.test 119
A.4.11.4 plt/test/scanner/pass/_assignment.out 119
A.4.11.5 plt/test/scanner/pass/_base_scanner.test 119
A.4.11.6 plt/test/scanner/pass/_base_scanner.out 119
A.4.11.7 plt/test/scanner/pass/_comment.test 120
A.4.11.8 plt/test/scanner/pass/_comment.out 120
A.4.11.9 plt/test/scanner/pass/_conditionals.test 120
A.4.11.10 plt/test/scanner/pass/_conditionals.out 120
A.4.11.11 plt/test/scanner/pass/_control_flow.test 121
A.4.11.12 plt/test/scanner/pass/_control_flow.out 121

A.4.11.13 plt/test/scanner/pass/_delimiters.test 121
A.4.11.14 plt/test/scanner/pass/_delimiters.out 121
A.4.11.15 plt/test/scanner/pass/_function.test 121
A.4.11.16 plt/test/scanner/pass/_function.out 121
A.4.11.17 plt/test/scanner/pass/_identifier.test 122
A.4.11.18 plt/test/scanner/pass/_identifier.out 122
A.4.11.19 plt/test/scanner/pass/_literal.test 122
A.4.11.20 plt/test/scanner/pass/_literal.out 122
A.4.11.21 plt/test/scanner/pass/_main_function.test 122
A.4.11.22 plt/test/scanner/pass/_main_function.out 122
A.4.11.23 plt/test/scanner/pass/_matrix.test 122
A.4.11.24 plt/test/scanner/pass/_matrix.out 122
A.4.11.25 plt/test/scanner/pass/_misc.test 123
A.4.11.26 plt/test/scanner/pass/_misc.out 123
A.4.11.27 plt/test/scanner/pass/_mixed_arithmetic.test 123
A.4.11.28 plt/test/scanner/pass/_mixed_arithmetic.out 123
A.4.11.29 plt/test/scanner/pass/_types.test 123
A.4.11.30 plt/test/scanner/pass/_types.out 123
A.4.12 plt/test/scanner/scripts/build.sh 123
A.4.13 plt/test/scanner/scripts/clean.sh 124
A.4.14 plt/test/scanner/scripts/test.sh 124
A.4.15 plt/test/scanner/tokenize.ml 129
A.5 plt/.travis-ci.sh 130
A.6 plt/hello_world_demo.sh 131
A.7 plt/Makefile 131
A.8 plt/stdlib.cmat 132

1. Introduction

1.1 Motivation

CMAT aims to combine the ability to perform matrix manipulations and other such linear algebra operations with the imperative programming style of C while removing the necessity for low-level memory management. CMAT is inspired by C and MATLAB, taking the best parts of both to produce a language with high versatility. As such, CMAT compiles down to LLVM IR; LLVM is a cross-platform runtime environment which allows CMAT code to work on any system as long as there is an LLVM port for it, which includes Windows, Mac OS X, and Linux or various processor architectures such as x86, MIPS, and ARM.

Ideally, we want to allow easy, efficient computation and matrix operations without sacrificing the structure of a full programming language. The goal was to build the potential for vector and matrix manipulation. Some other potential applications of our language include finding eigenvalues and eigenvectors, finding the inverse of a matrix, performing linear transformations on vectors, and solving numerical methods.

1.2 Background

CMAT is an imperative programming language that aims to provide a solid framework for vector and matrix manipulations. Matrices and vectors (more generally n-rank tensors; however, this language only supports 1st and 2nd rank) are fundamental structures widely used throughout math, computer science, and physics to represent data and mathematical equations.

Because we wanted to focus on numeric vector and matrix manipulation, we put our energies primarily toward vector/matrix manipulation. CMAT does not allow things such as function declarations within function declarations or dynamic memory allocation controlled by the user. All variables are allocated on the stack and are handled by an automatic garbage collector at the end of their scope.

2. Tutorial

2.1 Setting Up the Environment

The CMAT compiler has been built and tested using an Ubuntu 16.04 virtual machine. To create and manage our virtual machine, we used Virtualbox. The ISO image with OCaml and LLVM is https://courseworks2.columbia.edu/courses/10787/files/673708/download.

To log into the VM use the following credentials:

User:	plt4115
Password:	plt4115

Next, when the VM window is running, select Devices -> Network -> Connect Network Adapter.

Certain packages must be installed for the virtual machines. The following commands should be executed:

```
> sudo apt-get update
> sudo apt-get install ubuntu-desktop
```

Lastly, open the terminal and go to the directory where you wish the compiler files to be added. Enter the following command to add the compiler files:

```
> git clone https://github.com/frankcabada/plt.git
```

2.2 Building and Using the Compiler

2.2.1 Building

Assuming that you have the requisite versions of LLVM, OCaml, and opam installed (see README included in the top level of the plt git repository for these specific versions) and that you have cloned the git repository indicated above, you are ready to build the compiler! Simply navigate into the plt directory created from the clone and run the make command. If all of the prerequisite software is installed, the following is an example set of commands to build the compiler:

```
> cd plt
> make
```

This will compile all of the OCaml modules described in §5 into the compiler, `cmat.native`.

2.2.2 Usage

The CMAT compiler supports 3 different compilation flags:

Flag	Description
`-a`	Prints a representation of the program's AST
`-l`	Dumps the LLVM module created by this program to stdout
`-c`	Compiles the given source files into a .ll file

If you have a well-formed CMAT program, `foo.cmat`, the command to compile it into a program `foo.ll` is:

```
> ./cmat.native -c foo.cmat foo.ll
```

If the compile command (`./cmat.native -c`) is not given a name for the .ll file, it will output the LLVM module to a default filename of out.ll.

2.3 The Basics

2.3.1 Primitives

All primitives are declared by stating the type followed by its identifier. CMAT supports the following primitives:

- int
- bool
- float
- String
- vector
- matrix

```
int main() {
      int a; /* Declaring an integer variable */
      a = 3; /* Initialize on separate line */
      return 0;
```

```
}
```

2.3.2 Global Variables

Global variables can be accessed within the scope of the entire program. Because of this, global variables can be accessed within any function. Global variables are optional but must be declared at the very top of a CMAT program above the include statement and any functions. They must be assigned within a function. A global variable identifier is unique and the same declaration of the identifier cannot be used within a function for a local variable.

```
int x;     /* Global variable x */
int main() {
     x = 5;
     return 0;
}
```

2.3.3 Include Statement

CMAT has access to a standard library of functions with the use of a precompiler directive denoted at the top of the program. The library contains math, vector, and matrix functions.

```
#include <stdlib.cmat>;    /* Include statement */
int main() {
     return 0;
}
```

2.3.4 Operators

CMAT supports the following binary operators:

- Arithmetic (+ , - , * , /)
- Relational (== , != , < , <= , > , >=)
- Logical (&&, ||)

CMAT supports the following unary operators:

- Logical negation (!)
- Negate number (-)
- Increment/Decrement (++ , --)

```
int main() {
     int a;
     bool b; bool c;
```

```
    float f;

    a = 3;
    f = 5.5;
    ++a;       /* a is now 4 */
    a = a +4;  /* a is now 8 */
    f = f + 1; /* 1 is casted to a float, f is now 6.5 */

    b = true; c = false
    b == b; b != c; 4 > 2; /* These expressions evaluate to true */

}
```

2.3.5 Vectors and Matrices

CMAT makes it easy to work with vectors and matrices. Vectors are simply a 1-D matrix and matrices are 2-D. They can hold either integers or floats.

```
int main() {
    matrix int [3,3] mi1;           /* Matrix declaration */
    matrix int [3,3] mi2;
    matrix int [3,3] mi3;
    vector float [3] vf1;           /* Vector declaration */
    vector float [3] vf2;
    vector float [3] vf3;

    mi1 = mi2 = [1,2,3;4,5,6;7,8,9]; /* Matrix initialization */
    mi3 = mi1 + mi2;                /* Matrix addition */
    mi1[1,1]                        /* Matrix element access */

    vf1 = vf2 = |1.1|2.2|3.3|;      /* Vector initialization */
    vf3 = vf1 - vf2                 /* Vector subtraction */
}
```

2.3.6 Control Flow

Generally statements are executed in order unless a control flow statement is seen. Control flow statements allow for certain blocks of code to be executed, sometimes multiple times, if a condition is met. As a result branching and looping is allowed in CMAT. This section will further explain branching statements (if-then, if-then-else if-then, if-then-else, if-then-else if-then-else) and looping statements (while, for) supported by CMAT.

Branching

```
int main() {
      int x;
      int y;

      if (false) {     /* Program will not go through if statement */
            x = 5;
      }
      else {
            x = 10;
      }
      /* x is now 10 */
      y = 2;
      if (x == 10 && y == 2) {
            print_string(“Inside if statement”);
            /* Program will reach this point */
      }
}
```

Loops

The only looping statements supported in CMAT are ‘for’ and ‘while’ loops. The for loops allows a certain block of code to be executed multiple times as the program iterates over a range of values. The while loop executes a block of code multiple times as long as a condition is met.

```
int main() {
      int i;
      int x;
      x = 0

      for (i = 0; i < 5; i++) {
            x = x + 2;
      }
      /* x is now 10 */

      while (i < 10) {
            x = x + 2;
            i++;
      }
      /* x is now 20 */
}
```

2.3.7 Defining Functions

CMAT supports functions that return a data type (int, bool, float, String, vector, matrix) or void functions that do not return a value. Functions can accept arguments that are computed in an applicative order.

```
int main() {
      int i;
      i=4;
      print_int(foo(i)); /* Function call, prints 10 */
      return 0;
}

int foo(int j) { /* Function declaration */
      int i;
      i = j + 6;
      return i; /* Returns i with value 10 */
}
```

3. Language Reference Manual

3.1 Types

3.1.1 Primitive Types

int

An int is stored as a 32-bit signed integer. It can hold a range of values from -2,147,483,648 to 2,147,483,647. Declared and assigned as follows:

```
int a;
a = 5;
```

bool

A bool is stored as a 1-bit value. It can be either true (1) or false (0). Declared and assigned as follows:

```
bool b;
b = true;
```

float

A float is stored as a 64-bit floating point number. Floats require a decimal as well as a number after. The range is $\pm 2.23 \times 10^{-308}$ to $\pm 1.80 \times 10^{308}$. Precision for floats is 15 decimal places. Declared and assigned as follows:

```
float c;
c = 4.2;
```

null

A value of null is given to represent an absence of data.

String

Strings are represented as a sequence of characters. Double quotes (“ “) are used to define a string. Declared and assigned as follows:

```
String s;
s = “Hello World!”;
```

Vector

Vectors are like a 1-D matrix. A vector can contain integers and floats. Declared and assigned as follows:

```
vector int [4] vi;
vi = |1|2|3|4|;
```

Matrix

A matrix is an expanded version of a vector. The biggest matrix CMAT supports is 2-D. A matrix can contain integers and floats. Declared and assigned as follows:

```
matrix int [2,3] mi;
mi = [1,2,3;4,5,6];
```

3.1.2 Type Casting

Automatic type casting is supported during arithmetic operations between integers and floats. Before the operation is executed, the integer will be converted into a float. Example of automatic int -> float casting:

```
int a;
float b;
a = 3;
b = 5.5;
b = b + a;
```

3.2 Lexical Conventions

3.2.1 Identifiers

Identifiers in CMAT are case sensitive strings that represent variables and functions. Identifiers should always start with a letter followed by any arrangement of ASCII letters, digits, and the underscore character '_'.
There are certain identifiers that cannot be used as they are reserved for keywords and an error will occur if an identifier has the same spelling as these keywords. Example identifiers:

```
int my_num;
bool Flag1;
```

3.2.2 Keywords

Keywords are reserved identifiers for use in the CMAT language. An identifier with the same name cannot be created. CMAT recognizes the following keywords:

`if`	`else`	`for`	`while`
`int`	`float`	`bool`	`String`
`main`	`return`	`void`	`null`
`true`	`false`		

3.2.3 Operators

CMAT consists of the following operators. Integers and floats can perform arithmetic operations, including with each other due to automatic typecasting from int to float. Logical operators are strictly for booleans. Relational operators are open to integers, floats and some (== , !=) for booleans. However integers and floats cannot be used interchangeably when performing relational operations. Increment and decrement are only used by integers.

Operator	Name
=	Assign
==	Equal to
!=	Unequal to
>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal
+	Addition
-	Subtraction

*	Multiplication
/	Division
++	Increment
--	Decrement
&&	Logical AND
||	Logical OR
!	Logical NOT

Vectors and matrices have their own subset of operators. When performing operations, such as multiplication, with vectors, the vector must be on the right hand side of the operator.

Vector/Matrix Operator	Description
`+, -, *`	Matrix/Vector and scalar arithmetic operations
`v[x]`	Access specific element in vector v
`m[x,:]`	Access specific matrix row in matrix m
`m[:,y]`	Access specific matrix column in matrix m
`m[x,y]`	Access specific matrix element in matrix m

3.2.4 Precedence

The precedence of operators from highest to lowest is as follows:

```
!
* /
+ -
< > <= >=
== !=
&&
||
```

=

3.2.5 Comments

The characters (* begin a comment that terminates with the characters *). Example comment:

```
/*This is a comment*/
```

3.3 Syntax Notation

3.3.1 Expressions

Primary Expressions

Primary expressions are the most basic expressions which build to make more complex expressions. These include identifiers, constants, strings, or expressions surrounded by parentheses.

Postfix Expressions

In postfix expressions, the value of the expression is evaluated expression. The following are postfix expressions included in CMAT:

```
expression[expression]
expression(parameter-list)
```

Prefix Expressions

The operator is placed before the expression. In this way, the expression is altered by the operator before the value is used anywhere else. Examples of prefix expressions are increment and decrement:

```
int x;
x = 1;
++x;      /* x is 2 */
--x;      /* x is 1 */
```

Vector and Matrix References

Vector and Matrix elements are referenced through postfix expression of the form:

```
expression[expression]
```

The first expression is the identifier of an initialized vector/matrix. The second expression is either an integer for a vector or a pair of comma separated integers for a matrix. The pair can also include a colon (reference §3.2.3). Example matrix reference:

```
vector int [4] vi;
matrix int [2,3] mi;
vi = |1|2|3|4|;
mi = [1,2,3;4,5,6];
vi[1];    /* 2 */
mi[0,2];  /* 3 */
```

Function Calls

Function calls are postfix expressions of the form:

```
expression(parameter-list)
```

The first expression is a function identifier and the optional parameter-list consists of comma-separated expressions that are passed as the function parameters. Example function call for function `foo`:

```
foo(int x, int y);
```

3.3.2 Declaration

Type Specifiers

In CMAT, type specifiers define the type of an identifier. This indicates what type a variable holds, or in the case of a function what type it returns. Example declarations:

```
int x;
float f;
bool b;
String s;
```

Vector Declarations

A vector is declared similarly to other types (int, float) but with added information. The `vector` type is used along with a positive integer in brackets to denote the size. The last parameter is the identifier. Example vector declaration:

```
vector int [5] vi;
```

Matrix Declarations

A 2-D matrix has a similar declaration to vector. The `matrix` type is used along with a pair of two positive integers to denote the numbers of [rows, columns] the matrix is. The last parameter is the identifier. Example matrix declaration:

```
matrix float [3,3] mf;
```

Function Declarations

The format is as follows:

```
T name (T arg, …) { statements }
```

A function declaration starts with a return type. Then, an identifier is chosen followed by the formal arguments in parentheses if any. These arguments include the datatype and are comma-separated.

Example function declarations:

```
/* Functions */
void foo(int x){};
int bar(float f, int x){return 0;};
```

3.3.3 Initialization

An identifier is declared and initialized separately. These operations cannot be done at the same time. The '=' symbol is used to assign the value on the right to an identifier on the left.

int, float, bool, String

Example initialization using variables defined in §3.3.2:

```
x = 2;
f = 5.5;
b = true;
s = "Hello World!"
```

vector, matrix

Matrices are initialized within brackets and uses commas to separate columns and semicolons to separate rows. A vector is initialized within bars and uses bars to separate values. Example initialization using variables defined in §3.3.2:

```
vi = |1|2|3|4|5|;
mi = [1,2,3;4,5,6;7,8,9];
```

3.3.4 Statements

Expression Statement

Expression statements consist of standalone expressions which are executed before continuing to the next statement. An expression statement in CMAT is of the form:

```
expression;
```

These expressions are usually assignments or function calls. Expression statements may be empty if represented only by a semicolon.

Compound Statement

A compound statement (also known as a “block”) consists of several statements that can be used where a single statement is expected. Compound statements are of the form:

```
{ declaration-list statement-list }
```

declaration-list and statement-list are both optional meaning that it is possible to have an empty compound statement. Variables declared within compound statements do not live outside of that “block”.

Selection Statement

A selection statement chooses a specific flow to follow based on whether or not a condition is met. In CMAT, selection statements include `if` and `else` statements in the following forms:

```
if (expression) statement
if (expression) statement else statement
```

`expression` must be of `bool` or arithmetic type so that the program can evaluate whether or not a condition is met. The `if` statement is executed when `expression` does not evaluate to `false`, `null`, or 0. Otherwise, the statement within `else` is executed. An `else` cannot stand by itself and when used, it is paired with the last encountered `else`-less `if` at the same block nesting level. Example:

```
if (x == 2) {
    /* Do something */
}
else {
```

```
    /* Do something else*/
}
```

Iteration Statement

Iteration statements are used for looping in the following forms:

```
while (expression) statement
for (expression; expression; expression) statement
```

For `while` loops, statement is executed as long as expression meets the same conditions required for an `if` statement to be executed. Example:

```
while (flag1 == true) {
    /* Do something */
}
```

For `for` loops, the three expressions within the parentheses specify the number of iterations to loop over. The first expression initializes a value of any type. The second expression is evaluated in the same manner as an `if` condition. The `statement` will continue to be executed as long as this condition is met. The third expression is evaluated after the current iteration is executed in order to re-initialize the loop. Example:

```
for (x = 0; x < 10; x++) {
    /* Do something */
}
```

3.4 Standard Library Functions

3.4.1 Math

Function	Result
`int powi(int i, int n)`	Returns the nth power of i (int)
`float powf(float f, int n)`	Returns the nth power of i (float)
`int mod(int i, int n)`	Returns i modulo n

3.4.2 Vectors

Function	Result
`int dotiX(vector int [X] v1, vector int [X] v2)`	Returns the dot product of two integer vectors. Supports vectors of equal size. X can be 2 - 9
`float dotfX(vector float [X] v1, vector float [X] v2)`	Returns the dot product of two float vectors. Supports vectors of equal size. X can be 2 - 9

3.4.3 Matrix

Function	Result
`float sin(float x)`	Returns sin of angle. X is in degrees
`float cos(float x)`	Returns cos of angle. X is in degrees
`matrix float[2,2] rotation_2D_mat(float theta)`	Return 2 dimensional rotation matrix from an angle theta
`matrix float[3,3] rotation_3D_mat(float theta, vector int[3] axes)`	Return 3 dimensional rotation matrix from an angle theta and an axes vector

3.4.4 I/O

Function	Result
`void print_string(String s)`	Prints string s to stdout followed by a new line
`void print_int(int i)`	Prints int i to stdout followed by a tab

`void print_float(float f)`	Prints float f to stdout followed by a tab
`void printiX(vector int [X] V)`	Prints int vector of size X. X can be 2 - 9
`void printfX(vector float [X] V)`	Prints float vector of size X. X can be 2 - 9
`void printiXY(matrix int [X,Y] M)`	Prints int matrix of size XY. X and Y can be 2-9
`void printfXY(matrix float [X,Y] M)`	Prints float matrix of size XY. X and Y can be 2-9

3.5 Semantics

In CMAT, every statement must end with a semicolon ';'. Code blocks in control flow statements (if, else, for while) must always be enclosed in braces. Braces provide a more visual understanding of scope.

The program begins with a main function in a file. The main function must always have a return type of int. Main calls other functions defined which in turn may call other functions. When a function is called, the number actuals must match the number of format arguments in the function declaration. If a function has a return type, the end of the function must return the type specified in the function declaration. If a return object from a function is begins tored in a variable, the variable type must match the type of the return object.

4. Project Plan

4.1 Project Overview

In the planning stages of our project we set up a general outline on when to work on each module (scanner, parser, semant, etc.) in a timely manner throughout the semester. Because we did not have any previous experience in writing OCaml or compilers, we began with the scanner by setting up a basic framework inspired by the Micro C language. We first read through the code as a team and decided what would be similar and what would have to be completely different to fit our language.

Once we had a general idea of what we needed to do, we split up sections and worked in pairs or individually in order to implement each CMAT feature. Once we had a "completed" and compiling scanner we created tests to check that it produced the output we expected. These tests simply compared the scanner's output file with a file we wrote that contained the output we expected. The results of these tests pointed out where we needed to make changes or additions in our scanner until it passed all our tests. We then repeated this process throughout our project for the parser as well as every other part of our language compiler. As our tests revealed bugs or inconsistencies in CMAT, we found and patched them accordingly.

4.2 Project Timeline

- ***September 8, 2016***: First meeting; got to know each other and began brainstorming about possible focuses of our languages. Set up GroupMe for communication, setup Google Calendar for PLT events, set up Google Drive for sharing of deliverables (proposal, LRM, etc.), and set up Github repository for our project.
- ***September 15, 2016***: Second brainstorm meeting; determined we wanted to create a matrix-focused language with C-like syntax, possibly for game-building. Decided on many characteristics of the language such as data types, primitives, and operators. Set a group meeting times (every Tuesday, Thursday, and Sunday). Determined project roles.
- ***September 18, 2016***: Reviewed starred project proposals of past groups to determine how to begin formatting our own. Started working on our project proposal.
- ***September 22, 2016***: Continued work on our proposal.
- ***October 6, 2016***: First meeting with our assigned TA (Alexandra Medway). We met with Alexandra every Thursday since this meeting. Determined that matrices would be at most 2-dimensional, that we wanted to use static typing, and that we wanted to implement automatic garbage collection.
- ***October 9, 2016***: Reviewed the C LRM as well as the LRMs of previous projects in order to begin creating and formatting our own. We also worked on pushing our

scanner and parser closer to completion (goal was to complete our scanner and parser before the LRM due date).

- ***October 11, 2016***: Pushed scanner and parser forward. Gathered questions for Alexandra for our next TA meeting.
- ***October 13, 2016***: Meeting with Alexandra. Determined we should use Travic CI for continuous integration testing (integrates seamlessly with Github). Asked her how to test our scanner without a parser.
- ***October 18, 2016***: Set up Travis CI for testing. Continued work on scanner and parser. Started work on AST.
- ***October 20, 2016***: Meeting with Alexandra. Received answers to questions about compiler structure and Travis CI workflow.
- ***October 23, 2016***: Scanner completed, continued pushing parser and AST toward completion. Worked on LRM as well. Continued trying to fiddle with Travis's settings.
- ***October 25, 2016***: Put finishing touches on LRM due next day. Got Travis CI completely working and began writing scanner and parser tests.
- ***October 27, 2016***: Meeting with Alexandra. Received answers to questions about variable declaration sequencing and next steps after finishing up scanner, parser, AST.
- ***November 1, 2016***: Added more scanner and parser tests. Made changes to parser.
- ***November 2, 2016***: Meeting with Alexandra. Received feedback on our LRM.
- ***November 8, 2016***: Pushed parser forward and added parser tests.
- ***November 10, 2016***: Meeting with Alexandra. Received answers to questions about the SAST, LLVM, error checking, and the semantic analyzer.
- ***November 13, 2016***: Started SAST, semantic analyzer, utils and exceptions. Some fixing of older errors.
- ***November 15, 2016***: Continued work on SAST and semant. Began work on code generation and top level compiler which puts all the pieces together, cmat.ml. Started to work on infrastructure necessary for Hello World.
- ***November 17, 2016***: Meeting with Alexandra. Received answers to a few questions about Hello World. Pushed SAST, semant, and codegen forward.
- ***November 20, 2016***: Pushed SAST, semant, codegen forward again and finished Hello World.
- ***November 28, 2016***: Meeting with David Watkins. Received answers regarding LLVM intricacies, reserved function checking, and how programs should be built.
- ***December 1, 2016***: Meeting with Alexandra. Received feedback on Hello World and discussed final report.
- ***December 4, 2016***: Started working on final report. Continued to push semant and codegen forward.
- ***December 8, 2016***: Meeting with Alexandra. Received feedback on semant.
- ***December 11, 2016***: Continued working on final report. Nearly finished semant, leaving only codegen and a bit more testing.
- ***December 13, 2016***: Continued work on final report and codegen.

- ***December 14, 2016***: Continued work on final report and codegen. Started final presentation.
- ***December 15, 2016***: Continued work on final report and codegen. Prepare for final presentation.
- ***December 16, 2016***: Continued work on final report and codegen. Prepare for final presentation.
- ***December 17, 2016***: Putting finishing touches on codegen. Searching for bugs, inconsistencies. Still pushing final report and final presentation forward.
- ***December 18, 2016***: Preparing for final presentation.
- ***December 19, 2016***: Final presentation.

4.3 Challenges and Changes

The first challenge our team faced with this project was the constant struggle of having to learn how to actually begin piecing things together because much of the helpful content taught in class was usually covered after we had already started working on something. We had to learn mostly by deciphering the different parts of Micro C and previous class projects in order to figure out how we would structure our language until we eventually realized more or less what was going on. For this reason, some of our basic language functionality is similar to Micro C such as having to declare variables at the beginning of a function block. This is also why our language does not allow variable declaration and initialization on the same line. We decided that we would focus on implementing a basic version of all of CMAT's functionality first (with a strong emphasis on matrices) in order to better understand how we would apply this change in a future version.

The biggest challenge we faced was implementing CMAT matrices. Matrices are not a part of Micro C (or C) so we could not reference these languages as much for help. Implementing Binops with CMAT matrices was especially difficult because of how unique these operations are to matrices. A lot of time was spent figuring out how to access the appropriate values at each step of each operation and then properly forming the result. Additionally, we realized that index error checking for matrices had to be done in codegen because this required accessing matrix values. This was complicated further by the fact that some elements were as llvm values while others were not. These challenges took much of our time so we decided to simplify some things. More specifically, a major change to matrices in our language is the form of initialization. As of now, matrices can only be initialized by explicitly writing out all of its contents (i.e. m = [1,2,3,4: 5,6,7,8]) rather than using the form [x: y: z] to initialize a matrix with default values. However, with more time, this is definitely something we should implement.

Another major change to CMAT is the method of memory allocation. We were initially allocating everything on the stack as Micro C does for the sake of simplicity so that we could work on other language features in parallel. However, for most of the project we looked into automatic garbage collection and attempted to implement it in CMAT. Unfortunately, we ran into several complications with garbage collection and could not figure out how to implement it within our language. In order to implement it properly we would have to restructure our entire codegen file but by this point we had too much working functionality in it and we did not have enough time to carefully make all the necessary changes.

Instead, we attempted implementing dynamic memory allocation so that users would at least have the option to allocate memory on the heap with the *new* and *free* keywords. Although we were able to incorporate these keywords into our language, we had difficulties getting *free* to work properly. It would return an error because we were not accessing the correct pointer. This was related to the structure we have set up for variable declaration which again would mean the complete refactoring of a feature which we did not have enough time for. In the end we decided to completely remove the *new* and *free* keywords. These persistent structural issues showed us that we constrained ourselves in the way that we initially set up our language. We did not realize it early enough mainly because of our initial lack of knowledge and our inexperience in building an entire programming language.

4.4 Roles and Responsibilities

4.4.1 Roles

- Manager: Frank Cabada
    - Outlined project plan, team meetings, and task distribution
    - Led writing initial functionality in codegen for Hello World deliverable and focused on automatic garbage collection for final project (but unsuccessful) :(
    - Wrote compiler fail tests & CMAT program examples
- Language Guru: Michael Berkowitz
    - Led in language design decisions
    - Implemented major key features such as matrix functionality
    - Wrote several language tests
- System Architect: Marissa Ojeda
    - Led setup of scanner, parser, ast, sast and semantic analyzer
    - Implemented several necessary changes to each part of the compiler as language design changes made or features added
    - Focused on memory allocation functionality
- Tester: Daniel Rojas

- Outlined necessary pass/fail tests and delegated test creation
- Ensured continuous integration tool (Travis CI) properly set up and running with test suite
- Implemented fundamental language features in codegen such as expressions, assignments, etc.

4.4.2 Team responsibilities

All team members attended most meetings (of course, occasional legitimate absences are unavoidable in a project of this scale and length) and worked on different parts of the project outside of their specified roles. Most of the initial work was done together (struggled together on one or two computers in order to learn as a team) and then we split up specific coding tasks as well as final report sections.

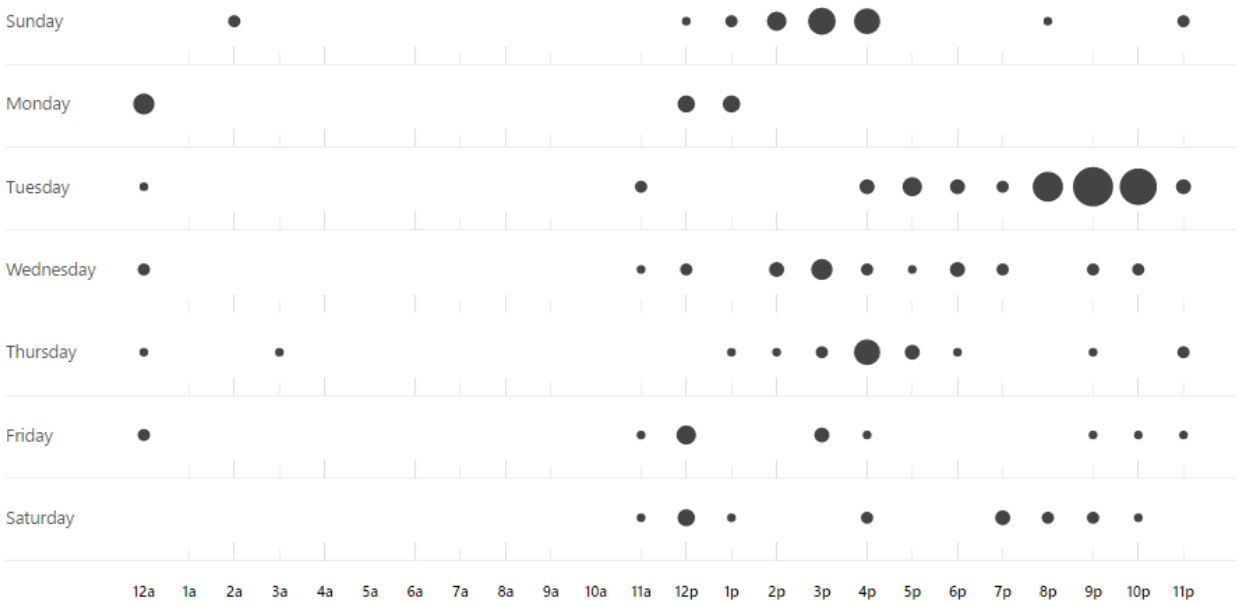
4.5 Development Environment

As we were provided a virtual machine image from Professor Edwards, we standardized our development environment to be in line with those standards. We used the following software versions:

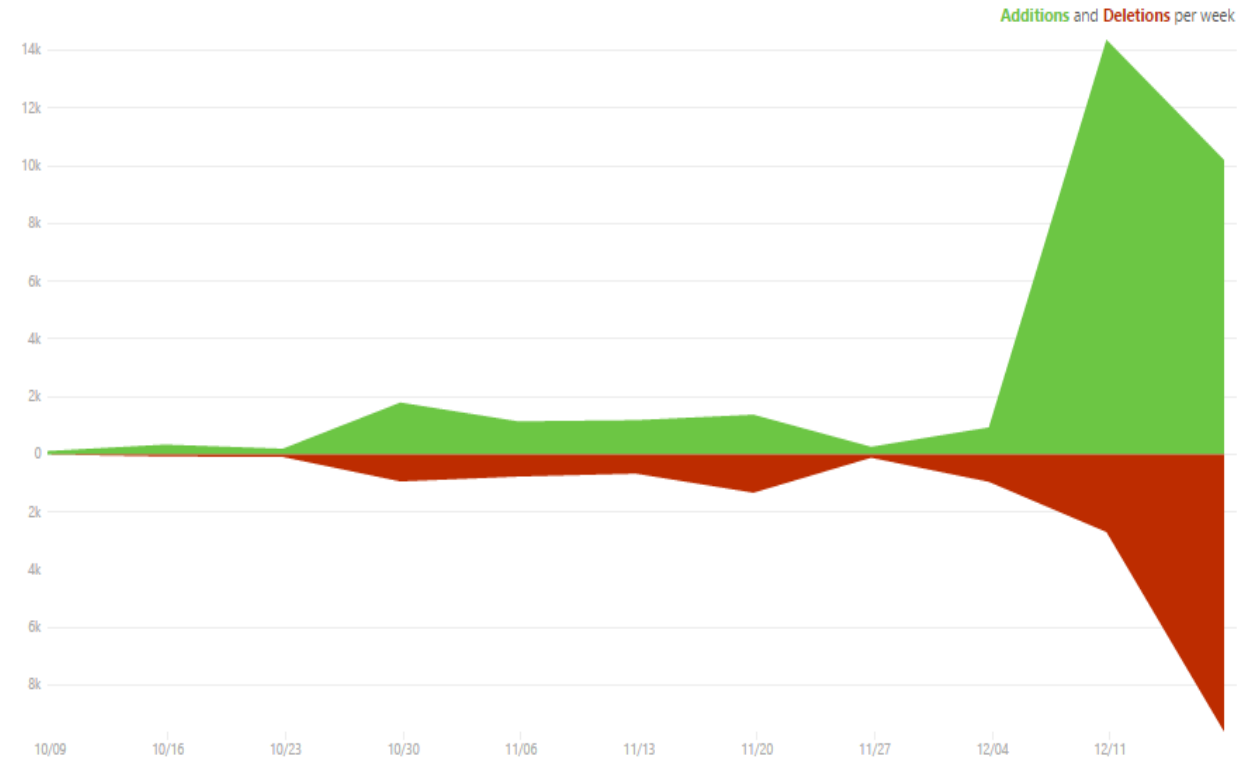
- **Ubuntu 16.04** - One of the most user-friendly Linux distributions with all LLVM and OCaml software readily accessible through apt-get. It is also the latest stable version of Ubuntu. The PLT official virtual image was used in Virtualbox by all members of our group to ensure hardware consistency during the development process.
- **LLVM 3.0.8** - The latest version of LLVM and included with the PLT virtual image.
- **GroupMe**: Group message application used for all official group communications.
- **Google Drive**: Used to store all deliverables (proposal, LRM, this very document that you are reading) while allowing us to simultaneously work on different parts of assignments.
- **Github**: Version control system used to maintain consistency among our source code. We were very careful to communicate through GroupMe and during meetings when it was necessary to merge master-level changes into our own respective branches.
- **Sublime/Atom**: Both open-source text editors. We did not feel it necessary to require exact standardization in this area.

4.6 Version Control Statistics

4.6.1 Punch Card



4.6.2 Code Frequency



4.7 Master Branch Commit History

```
Sun Dec 18 14:46:42 2016 -0500 - 771fb13: adding top-level Makefile - meberko
Sun Dec 18 02:07:36 2016 -0500 - 040064a: oops committed an ll file - meberko
Sun Dec 18 02:06:17 2016 -0500 - 00bd426: adding transpose and new matrix and vector tests - meberko
Sat Dec 17 22:10:15 2016 -0500 - fa3aa6d: matrix*vector multiplication DONE. FUNCTIONALITY ALL DONE - meberko
Sat Dec 17 21:45:32 2016 -0500 - d8912d1: removing non-exhaustive warnings - meberko
Sat Dec 17 21:33:10 2016 -0500 - 533e985: matrix_row and matrix_cols working, let's gooooooooo - meberko
Sat Dec 17 20:35:09 2016 -0500 - a057273: Merge branch 'master' into mike-new - meberko
Sat Dec 17 20:26:00 2016 -0500 - c1ea426: Wrote fibonacci.cmat program, removed logical operators test -
frankcabada
Sat Dec 17 20:19:06 2016 -0500 - 6ac2fc7: Merge branch 'master' into mike-new - meberko
Sat Dec 17 20:15:46 2016 -0500 - 2e062ee: quick fix of stdlib - meberko
Sat Dec 17 19:45:57 2016 -0500 - 58e58bc: Removed not operator for ints - Daniel Rojas
Sat Dec 17 19:41:54 2016 -0500 - 9678d6f: Merge branch 'master' into mike-new - meberko
Sat Dec 17 19:41:26 2016 -0500 - 336fefa: MATRIX MULTIPLICATION WORKING WOOOOOO - meberko
Sat Dec 17 19:12:26 2016 -0500 - 0c6cbde: Merge branch 'master' into marissa_gc - Marissa Ojeda
Sat Dec 17 19:08:26 2016 -0500 - edbfe14: added vector_lit to codegen and changed parser for vector lit and free -
Marissa Ojeda
Sat Dec 17 16:53:58 2016 -0500 - 7bbd5c7: Removed unusued match case - frankcabada
Sat Dec 17 16:45:11 2016 -0500 - ed60c2c: Merge branch 'master' into fcabada-tests - frankcabada
Sat Dec 17 16:44:30 2016 -0500 - f86fad1: New fail tests for compiler - frankcabada
Sat Dec 17 13:59:04 2016 -0500 - 91a7faf: num*vector and num*matrix working! tests added. We flyin' - meberko
Sat Dec 17 12:56:21 2016 -0500 - 863a5f5: Added Snew and Sfree to semant - Marissa Ojeda
Sat Dec 17 12:47:41 2016 -0500 - 60a9b44: solved merge conflict - Marissa Ojeda
Sat Dec 17 12:44:45 2016 -0500 - b6d09d9: Added new and free to semant - Marissa Ojeda
Sat Dec 17 12:24:44 2016 -0500 - 0e7a2ad: Merge branch 'master' into mike-new - meberko
Sat Dec 17 12:21:16 2016 -0500 - d54fb5b: Adding new, free, and vector_lit to scanner, parser, ast, sast, semant,
utils - Marissa Ojeda
Sat Dec 17 12:16:54 2016 -0500 - 4bebdac: vector addition/subtraction working. tests added - meberko
Sat Dec 17 11:52:56 2016 -0500 - ae5a6a7: Fixed logical operators to work with ints, not just bools - Daniel Rojas
Fri Dec 16 23:48:52 2016 -0500 - b6604ea: Merge branch 'master' into mike-new - meberko
Fri Dec 16 23:48:21 2016 -0500 - 186685a: OMG MATRIX ADDITION AND SUBTRACTION WORK WOOHOOHOOOOOOO - meberko
Fri Dec 16 22:37:29 2016 -0500 - 7cec5a1: Tweaked print_string to allow printing string variables - Daniel Rojas
Fri Dec 16 21:54:11 2016 -0500 - c99d9b1: a lot of stuff. Attempting to implement matrix binops. Tough stuff.
stdlib added, can print matrices - meberko
Fri Dec 16 12:51:56 2016 -0500 - 1877685: adding include test as well - meberko
Fri Dec 16 12:41:12 2016 -0500 - 34916af: adding matrix literals and tests. have to implement array out of bounds
checking - meberko
Thu Dec 15 23:57:31 2016 -0500 - d74a0d4: Added proper compile command to hello_world_demo, removed garbage
collection stuff in codegen - frankcabada
Thu Dec 15 21:12:54 2016 -0500 - 628f21e: #include is now a thing! Changed all tests and test scripts
appropriately - meberko
Thu Dec 15 18:48:55 2016 -0500 - f3e69ab: removed Matrix_init and added Rows, Cols, Len - meberko
Thu Dec 15 16:48:42 2016 -0500 - 32efea2: vector/matrix accesses WORKING! added vector/matrix element assignment
to assign test - meberko
Thu Dec 15 13:26:04 2016 -0500 - 91d9d27: Merge branch 'master' into mike-new - meberko
Thu Dec 15 13:25:49 2016 -0500 - 53c53f7: fixed semant saying global vars are undefined, added global checking
into assign test - meberko
Thu Dec 15 03:00:10 2016 -0500 - 0485c63: Fixed issue caused by garbage collection, currently debugging StringMap
issues - frankcabada
Thu Dec 15 02:58:43 2016 -0500 - 28dd93a: Merge branch 'master' into fcabada-garbage-collection - frankcabada
Thu Dec 15 00:24:13 2016 -0500 - ebaa5c7: Fixed inc and dec - Daniel Rojas
Wed Dec 14 21:59:46 2016 -0500 - 0d43912: Removed list reverse code - frankcabada
Wed Dec 14 21:29:53 2016 -0500 - b3a7e9c: Added float and bool unops. Inc and Dec still not working - Daniel Rojas
Wed Dec 14 18:38:34 2016 -0500 - 0d6251e: Merge branch 'master' into mike-new - meberko
Wed Dec 14 18:31:52 2016 -0500 - 0d6e313: Implemented casting in codegen for int to float conversion - Daniel
Rojas
Wed Dec 14 18:16:22 2016 -0500 - 794cc83: crap I broke travis - meberko
```

```
Wed Dec 14 18:09:41 2016 -0500 - 100cd79: adding compiler tests into .travis-c.sh! Finally have compiler tests!
Also some semant fixes - meberko
Wed Dec 14 17:10:31 2016 -0500 - 39e5dc8: Merge branch 'master' into mike-new - meberko
Wed Dec 14 17:10:18 2016 -0500 - 5ad7fa9: working on compiler tests - meberko
Wed Dec 14 16:58:34 2016 -0500 - 2043d06: Fixed merge with binops - Daniel Rojas
Wed Dec 14 16:53:44 2016 -0500 - 5271b01: Merge branch 'master' of https://github.com/frankcabada/plt - Daniel
Rojas
Wed Dec 14 16:46:04 2016 -0500 - 20ea900: fixing hello world derp - meberko
Wed Dec 14 16:37:46 2016 -0500 - 4cec4e9: adding support for vector and matrix binops in semant - meberko
Wed Dec 14 15:54:38 2016 -0500 - 4b0c96a: Implemented binop for float separate from ints - Daniel Rojas
Wed Dec 14 15:13:12 2016 -0500 - 0fc10e8: Merge branch 'master' into mike-new - meberko
Wed Dec 14 15:08:47 2016 -0500 - 8a79bb4: added semantic checking for vector/matrix binops (almost done oops
forgot int/vectors int/matrix) - meberko
Wed Dec 14 15:06:51 2016 -0500 - a40a842: Merge branch 'master' into daniel_codegen - Daniel Rojas
Wed Dec 14 15:05:33 2016 -0500 - 7842ef2: Merge branch 'daniel_codegen' of https://github.com/frankcabada/plt into
daniel_codegen - Daniel Rojas
Wed Dec 14 15:05:15 2016 -0500 - 7a848fb: Fixed float llvm type - Daniel Rojas
Wed Dec 14 14:32:40 2016 -0500 - 6cd3899: removed else as stmt in ast and selse as sstmt in sast, removed them
from semant and utils too - Marissa Ojeda
Wed Dec 14 14:17:08 2016 -0500 - 69da787: deleted reversing list for build_function_body - Marissa Ojeda
Tue Dec 13 23:51:50 2016 -0500 - f57ef0b: solved merge conflicts - Marissa Ojeda
Tue Dec 13 23:49:43 2016 -0500 - 3aa480b: Added printing for int and float - Daniel Rojas
Tue Dec 13 23:42:40 2016 -0500 - 442dbc5: Merge branch 'daniel_codegen' of https://github.com/frankcabada/plt into
daniel_codegen - Daniel Rojas
Tue Dec 13 23:33:50 2016 -0500 - 406b6ac: reversed the list where build_function_body is called; created free_main
function that finds main and call free var on stringmap - Marissa Ojeda
Tue Dec 13 18:21:22 2016 -0500 - 480a632: Merge branch 'master' into mike-new - meberko
Tue Dec 13 18:17:06 2016 -0500 - 1d465c9: whoops forgot to change a test report - meberko
Tue Dec 13 18:15:50 2016 -0500 - 9e94333: reorganizing parser test folder and adding parser fail tests - meberko
Tue Dec 13 17:17:31 2016 -0500 - b16f6c6: reorganizing scanner test folder and adding scanner fail tests - meberko
Sun Dec 11 15:37:02 2016 -0500 - 8f86ab4: Implemented ignore return when freeing for garbage collection -
frankcabada
Sun Dec 11 15:35:04 2016 -0500 - 07ceca4: Merge branch 'master' into daniel_codegen - frankcabada
Sun Dec 11 15:17:57 2016 -0500 - 60ae953: deleted commented out code - Marissa Ojeda
Sun Dec 11 15:13:09 2016 -0500 - 7611449: Call new stringmaps functions created - Marissa Ojeda
Sun Dec 11 14:43:55 2016 -0500 - a4c2ca6: added vector and matrix generation to codegen! It worksgit status -
meberko
Sun Dec 11 14:26:50 2016 -0500 - 6cd5d41: Added stringmap for return variables and look up fxn - Marissa Ojeda
Sun Dec 11 13:36:57 2016 -0500 - c1ab0a0: fixing hello_world_demo.sh - meberko
Sun Dec 11 12:59:43 2016 -0500 - 286d8b7: adding some compiler tests; not done yet - meberko
Fri Dec 9 16:47:37 2016 -0500 - 42d8186: Added inc and dec - Daniel Rojas
Fri Dec 9 15:10:29 2016 -0500 - 3680a2f: Merge branch 'master' into daniel_codegen - Daniel Rojas
Fri Dec 9 12:25:26 2016 -0500 - 7b22c6f: gotta make sure travvy still works - meberko
Fri Dec 9 12:24:10 2016 -0500 - ef09c00: added ability to name .ll output files - meberko
Thu Dec 8 16:38:08 2016 -0500 - f8de64a: removing comment from semant - meberko
Thu Dec 8 16:37:06 2016 -0500 - 2387344: Merge branch 'daniel_codegen' of https://github.com/frankcabada/plt into
daniel_codegen - Daniel Rojas
Thu Dec 8 16:36:56 2016 -0500 - 850d952: Merge branch 'master' into mike-new - meberko
Thu Dec 8 16:33:17 2016 -0500 - 80a972a: fixing a parser test - meberko
Thu Dec 8 16:31:56 2016 -0500 - fd014da: Added floats to codegen - frankcabada
Thu Dec 8 16:25:54 2016 -0500 - bfbf3d7: Merge branch 'daniel_codegen' of https://github.com/frankcabada/plt into
daniel_codegen - Daniel Rojas
Thu Dec 8 16:19:18 2016 -0500 - 63c1915: Merge branch 'master' into daniel_codegen - frankcabada
Thu Dec 8 16:18:11 2016 -0500 - 9ad7868: Merge branch 'daniel_codegen' of https://github.com/frankcabada/plt into
daniel_codegen - Daniel Rojas
Thu Dec 8 16:16:21 2016 -0500 - 1624e49: Fixed issue with free_locals - frankcabada
Thu Dec 8 16:08:51 2016 -0500 - 0eb012c: some slight fixes which FIXED HELLO WORLD WOOOOOO - meberko
Thu Dec 8 15:54:44 2016 -0500 - dfff4e9: ALL non-exhaustive warnings gone from semant and utils. Added VECTOR
token, Vector_access, and SVector_access. Lookin' goody. - meberko
Thu Dec 8 15:50:28 2016 -0500 - 5c86bc4: Merge branch 'daniel_codegen' of https://github.com/frankcabada/plt into
daniel_codegen - Daniel Rojas
```

```
Thu Dec 8 15:47:41 2016 -0500 - c3ec26b: Initial code for garbage collection - frankcabada
Thu Dec 8 14:04:56 2016 -0500 - 389c89b: all non-exhaustive errors gone from semant - meberko
Thu Dec 8 13:26:30 2016 -0500 - ab33db0: merged semant with Marissa's semant, it compiles! - meberko
Wed Dec 7 22:15:32 2016 -0500 - 2aa7167: Added check_call, added fname_map argument to most functions - Marissa
Ojeda
Wed Dec 7 22:05:44 2016 -0500 - 7cf9993: almost all matrix stuff done! - meberko
Wed Dec 7 20:39:42 2016 -0500 - f17afe6: Merge branch 'daniel_codegen' of https://github.com/frankcabada/plt into
daniel_codegen - Daniel Rojas
Wed Dec 7 20:13:24 2016 -0500 - 5f355f5: Merge branch 'daniel_codegen' of https://github.com/frankcabada/plt into
daniel_codegen - Daniel Rojas
Wed Dec 7 19:52:27 2016 -0500 - a904860: bye bye break - meberko
Wed Dec 7 00:11:01 2016 -0500 - 090c668: revised Return --> SReturn - meberko
Wed Dec 7 00:04:13 2016 -0500 - 079426a: Merge branch 'master' into mike-new - meberko
Wed Dec 7 00:01:04 2016 -0500 - 6b65710: Merge branch 'Marissa-semant' into mike-new - meberko
Tue Dec 6 22:39:49 2016 -0500 - 72e9664: Started coding check_call - Marissa Ojeda
Tue Dec 6 22:05:54 2016 -0500 - b0551af: nearly done stmts in semant - meberko
Tue Dec 6 21:49:54 2016 -0500 - 5e92a9e: Merge branch 'master' into daniel_codegen - frankcabada
Tue Dec 6 21:49:00 2016 -0500 - c7345a5: Merge branch 'daniel_codegen' of https://github.com/frankcabada/plt into
daniel_codegen - Daniel Rojas
Tue Dec 6 21:46:12 2016 -0500 - de1c1ba: Added for loop to codegen & fixed issue of SExpr in sast - frankcabada
Tue Dec 6 21:16:49 2016 -0500 - 3a54b85: actual merge with mike-new - meberko
Tue Dec 6 21:08:13 2016 -0500 - 20d775c: Merge branch 'Marissa-semant' of https://www.github.com/frankcabada/plt
into Marissa-semant - meberko
Tue Dec 6 21:00:28 2016 -0500 - a3bdbcf: Block, If, While, For now checked - meberko
Tue Dec 6 20:59:48 2016 -0500 - 6c97861: Added check_assign and check_binop - Marissa Ojeda
Tue Dec 6 20:33:38 2016 -0500 - 7eba02c: Merge branch 'daniel_codegen' of https://github.com/frankcabada/plt into
daniel_codegen - Daniel Rojas
Tue Dec 6 20:33:19 2016 -0500 - 662caed: Now translating to sast, hello_world_demo does not compile - frankcabada
Tue Dec 6 20:19:31 2016 -0500 - 09f8361: Merge branch 'daniel_codegen' of https://github.com/frankcabada/plt into
daniel_codegen - Daniel Rojas
Tue Dec 6 20:19:04 2016 -0500 - aaa49ed: Merge branch 'master' into daniel_codegen - frankcabada
Tue Dec 6 20:16:26 2016 -0500 - 8e24310: fixing up semant for Frank - meberko
Tue Dec 6 19:55:50 2016 -0500 - f47fc86: Merge branch 'daniel_codegen' of https://github.com/frankcabada/plt into
daniel_codegen - Daniel Rojas
Tue Dec 6 19:46:47 2016 -0500 - f353389: Merge branch 'master' into daniel_codegen - frankcabada
Tue Dec 6 19:45:15 2016 -0500 - 5aefabc: Merge branch 'daniel_codegen' of https://github.com/frankcabada/plt into
daniel_codegen - Daniel Rojas
Mon Dec 5 00:16:05 2016 -0500 - 6923007: merged semant with mike-new - meberko
Sun Dec 4 16:01:03 2016 -0500 - 19a6cd1: Fixed merge conflict in sast. stmt list to sstmt list - Daniel Rojas
Sun Dec 4 15:56:45 2016 -0500 - 38dee07: Changed codegen to reference sast instead of ast. Added if and while -
Daniel Rojas
Sun Dec 4 15:46:57 2016 -0500 - 395af5f: Added check_unop, expr_to_sexpr, and get_id_type - Marissa Ojeda
Sun Dec 4 15:42:42 2016 -0500 - 111d0e5: set up infrastructure for checking stmts and converting stmt-->sstmt. So
far, only stmt thru all steps is Return - meberko


Wed Nov 30 12:26:57 2016 -0500 - f0702f4: stop attempts to merge after success - meberko
Wed Nov 30 12:21:45 2016 -0500 - fb8caaf: okay made some solid changes. Semant.check_functions now creates and
returns a sast to main cmat compilation. I fudged convert_fdecl_to_sfdecl just a bit. sbody needs to be changed. -
meberko
Wed Nov 30 11:08:20 2016 -0500 - 175e777: changing .travis-ci.sh to check output of hello world is actually
correct. - meberko
Tue Nov 29 22:23:36 2016 -0500 - b60bfb2: progress made on function checking - meberko
Tue Nov 29 21:19:48 2016 -0500 - 7fb560e: Semant.check_var_decls TOTALLY done - meberko
Tue Nov 29 20:22:22 2016 -0500 - 9b660ae: globals get checked now! - meberko
Tue Nov 29 16:06:42 2016 -0500 - 9645e75: adding first compiler test - meberko
Sun Nov 27 20:17:22 2016 -0500 - 3ae98ef: Merge branch 'master' of https://github.com/frankcabada/plt - Marissa
Ojeda
Sun Nov 27 20:15:35 2016 -0500 - c3c8f45: Added starting program point in semant let check - Marissa Ojeda
Mon Nov 21 13:41:28 2016 -0500 - f4db636: IT WORKS WOOOOOO - meberko
Mon Nov 21 13:32:40 2016 -0500 - 129b80a: ugh - meberko
Mon Nov 21 13:16:40 2016 -0500 - b0496a7: troubleshoot travvy - meberko
```

```
Mon Nov 21 13:06:08 2016 -0500 - a003141: I shall make no claims about the potential success of this build -
meberko
Mon Nov 21 12:53:55 2016 -0500 - 39786e2: okay I've said this before but this SHOULD work - meberko
Mon Nov 21 12:38:25 2016 -0500 - 11d13ad: sigh, maybe this will work - meberko
Mon Nov 21 12:33:00 2016 -0500 - 8c1dbad: messing with installations on travis - meberko
Mon Nov 21 12:25:45 2016 -0500 - 544b1d1: trying to add hello world to .travis-ci.sh - meberko
Sun Nov 20 16:46:22 2016 -0500 - 2b42100: changing .gitignore to not ignore clean.sh - meberko
Sun Nov 20 16:37:54 2016 -0500 - 418b393: OMG IT WORKS HELLO WORLD WORKS WOWOWOWOWOWOWOWOWOWOWOW - meberko
Sun Nov 20 16:17:17 2016 -0500 - 8e4c84a: fixing codegen - meberko
Sun Nov 20 15:33:17 2016 -0500 - d7679a5: Restructuring! - meberko
Sun Nov 20 15:10:21 2016 -0500 - 917dd0a: OMG IT COMPILES. IT LIVESSSSSSSSS. - meberko
Sun Nov 20 14:22:13 2016 -0500 - 6d6875a: Merge branch 'master' into mike-new - meberko
Sun Nov 20 14:22:08 2016 -0500 - bbbdfb2: codegen free of errors, working on cmat.ml - meberko
Fri Nov 18 15:04:32 2016 -0500 - dc43040: reconfiguring master to run tests. master can break now - meberko
Fri Nov 18 15:01:22 2016 -0500 - abd7552: fixing tests after main rehaul - meberko
Fri Nov 18 12:49:44 2016 -0500 - 701249b: fixed some stuffs. semant compiles; codegen still doesn't - meberko
Fri Nov 18 11:13:49 2016 -0500 - 2301f25: added functions to sast functions in semant - Daniel Rojas
Thu Nov 17 23:43:20 2016 -0500 - d232710: Removed main. Fixed parser, added global vars to ast and parser. -
Daniel Rojas
Thu Nov 17 16:59:18 2016 -0500 - fc05a41: Merge branch 'master' of https://github.com/frankcabada/plt -
frankcabada
Thu Nov 17 16:59:14 2016 -0500 - 901a90b: Copied most of MicroC codegen - frankcabada
Tue Nov 15 22:20:52 2016 -0500 - 4af305c: Added print_line and fixed few bugs in semant. Minor changes to sast,
ast, utils - Daniel Rojas
Tue Nov 15 21:46:22 2016 -0500 - c9de514: started codegen.ml and top-level cmat.ml; adding hello_world directory -
meberko
Sun Nov 13 16:57:01 2016 -0500 - 2edab64: fixing stuff we broke - meberko
Sun Nov 13 16:43:22 2016 -0500 - 52bf0d0: stole utils and exceptions from DICE. Pushed semant forward - meberko
Sun Nov 13 13:42:13 2016 -0500 - ba02b11: Sas.mli file updated - Marissa Ojeda
Wed Nov 9 19:53:15 2016 -0500 - 6e29e87: adding parser test for all expressions - meberko
Wed Nov 9 14:10:41 2016 -0500 - 26705d8: adding parser tests for assign, formal opts, primitive decls, and
statements - meberko
Tue Nov 8 20:28:12 2016 -0500 - df66bd9: Added num_lit to scanner, parser, ast, and fixed failing scanner tests -
Daniel Rojas
Tue Nov 8 19:45:58 2016 -0500 - ebf7bd0: Added comment to ast - Marissa Ojeda
Tue Nov 8 19:34:25 2016 -0500 - e03953f: parser testsgit status - meberko
Tue Nov 8 18:18:18 2016 -0500 - a0a899c: small changes - meberko
Tue Nov 8 17:22:03 2016 -0500 - 0b43d1c: rearrange so scanner, parser, ast are taken from top level for testing -
meberko
Tue Nov 8 11:23:50 2016 -0500 - 6b757f9: Added 7 tests to test/scanner - Marissa Ojeda
Tue Nov 8 11:19:49 2016 -0500 - 1a814e1: Added main function to ast/parser. Edited parserizer. Edited _test in
test/scanner - Marissa Ojeda
Fri Nov 4 15:05:56 2016 -0400 - e3d1973: Parserizer.ml edit - Daniel Rojas
Thu Nov 3 17:35:59 2016 -0400 - 26121fb: Implemented test color code for parser test script - frankcabada
Thu Nov 3 17:34:04 2016 -0400 - b5d69c6: Merge branch 'master' into fcabada-tests-parser - frankcabada
Thu Nov 3 17:33:03 2016 -0400 - f511775: Color coded pass and fail tests - frankcabada
Thu Nov 3 17:19:06 2016 -0400 - 1396285: fixed merge conflicts with master - Marissa Ojeda
Thu Nov 3 17:15:17 2016 -0400 - 8765f9f: Added scanner tests and modified scanner, parser, and ast within
test/scanner - Daniel Rojas
Thu Nov 3 16:33:47 2016 -0400 - 961d0a4: Commented out elseif - Marissa Ojeda
Wed Nov 2 15:49:42 2016 -0400 - 25f75ec: Added const and break - Marissa Ojeda
Wed Nov 2 15:24:52 2016 -0400 - e910338: Changed main back to program in parser and ast - Marissa Ojeda
Wed Nov 2 15:21:00 2016 -0400 - b907c5b: Added matrix decls and matrix literals to ast, parser, scanner. Changed
typ to primitives. Added Inc and Dec Uop. Added main instead of program for start. Added Null. Added matrix access
for elements/row/columns. - Marissa Ojeda
Wed Nov 2 00:07:02 2016 -0400 - 8232227: Changed double to float - Marissa Ojeda
Tue Nov 1 22:59:41 2016 -0400 - 9334872: Added cmi and cmo to gitignore - Daniel Rojas
Tue Nov 1 22:57:47 2016 -0400 - d31575b: Merge fix - Marissa Ojeda
Tue Nov 1 22:54:00 2016 -0400 - d75a137: Colon added to ast/parser/scanner - Marissa Ojeda
Tue Nov 1 22:52:21 2016 -0400 - be7d73a: Fixed formatting in ast - Daniel Rojas
Tue Nov 1 22:45:20 2016 -0400 - ca5aa06: Changed txt_of_expr to parserize - Daniel Rojas
```

```
Tue Nov 1 22:40:23 2016 -0400 - 388b570: ast - meberko
Tue Nov 1 22:36:37 2016 -0400 - 12e011a: making ast pretty - meberko
Tue Nov 1 22:26:32 2016 -0400 - 22edaa3: merging master into mike - meberko
Tue Nov 1 22:23:08 2016 -0400 - 303ac19: working on parserizer - meberko
Tue Nov 1 22:17:59 2016 -0400 - 2842e9b: Merge branch 'master' into marissa - Marissa Ojeda
Tue Nov 1 22:17:26 2016 -0400 - 0e3d251: Added strings and double to parser/scanner/ast - Marissa Ojeda
Tue Nov 1 21:41:44 2016 -0400 - c3597fc: Copied buildparser to top level directory - frankcabada
Tue Nov 1 21:36:30 2016 -0400 - 1fe0c8d: Merge branch 'master' into mike - meberko
Tue Nov 1 21:36:23 2016 -0400 - 272131d: parser tests - meberko
Tue Nov 1 21:33:53 2016 -0400 - ab82ea9: Deleted null - Marissa Ojeda
Tue Nov 1 21:32:14 2016 -0400 - fbb9525: Tests and fixed merge - Marissa Ojeda
Tue Nov 1 21:28:18 2016 -0400 - 8d5ea76: Merge branch 'master' into marissa - Marissa Ojeda
Tue Nov 1 21:25:53 2016 -0400 - 9d2b8b1: Added double to parser/ast - Marissa Ojeda
Tue Nov 1 21:23:05 2016 -0400 - 820da29: reorganizing parser - meberko
Tue Nov 1 21:05:24 2016 -0400 - a78a8c8: Stripped down parser, ast - Marissa Ojeda
Tue Nov 1 20:56:27 2016 -0400 - 46766e6: adding many scanner tests - meberko
Tue Nov 1 20:16:23 2016 -0400 - 7ab4188: Merge branch 'master' into mike - meberko
Tue Nov 1 20:12:03 2016 -0400 - 203e53c: Merge branch 'master' into mike - meberko
Tue Nov 1 20:10:23 2016 -0400 - 3647964: Merge branch 'master' into fcabada-test - frankcabada
Tue Nov 1 20:09:37 2016 -0400 - 644f8e0: Deleted resources directory with example scanner - frankcabada
Tue Nov 1 17:26:06 2016 -0400 - fb2032b: tests - meberko
Tue Nov 1 17:20:29 2016 -0400 - daf853b: trying to fix testing oops - meberko
Tue Nov 1 17:09:19 2016 -0400 - 660b654: Merge branch 'master' into mike - meberko
Tue Nov 1 17:03:07 2016 -0400 - f748960: adding parserizer.ml - meberko
Tue Nov 1 16:43:05 2016 -0400 - fb10e03: whoops fixing travis - meberko
Tue Nov 1 16:37:23 2016 -0400 - 1e055da: changing up testing directory structure - meberko
Tue Oct 25 00:17:25 2016 -0400 - 1e3a063: organized scanner, extended scanner test - meberko
Mon Oct 24 00:32:01 2016 -0400 - a266793: Travis actually running a test - meberko
Mon Oct 24 00:29:15 2016 -0400 - 41cf27a: travis - meberko
Mon Oct 24 00:20:49 2016 -0400 - 6ebadcd: testing travis shell - meberko
Mon Oct 24 00:18:28 2016 -0400 - b650968: testing travvy - meberko
Mon Oct 24 00:09:32 2016 -0400 - 7f8569d: travis cmon - meberko
Mon Oct 24 00:02:19 2016 -0400 - 07c1b06: travis workkk - meberko
Sun Oct 23 23:50:05 2016 -0400 - c62c04d: travis works! praise travis! actually running a scanner test - meberko
Sun Oct 23 23:38:07 2016 -0400 - e0e0277: testing travis - meberko
Sun Oct 23 16:41:17 2016 -0400 - 829b357: TEST PLZ WORK - meberko
Sun Oct 23 16:23:34 2016 -0400 - ae22bfe: changed travis again - meberko
Sun Oct 23 16:17:20 2016 -0400 - 78003cc: changing travis - meberko
Sun Oct 23 16:08:29 2016 -0400 - b582ab4: cleaned up testing - meberko
Sun Oct 23 14:48:27 2016 -0400 - 0aefd26: Added .gitignore - Daniel Rojas
Fri Oct 21 00:21:29 2016 -0400 - 16f1241: fixing travis - meberko
Fri Oct 21 00:17:00 2016 -0400 - d065fae: test working! just run ./make_and_run_test.sh. To clean, run
./makeclean.sh - meberko
Thu Oct 20 16:55:52 2016 -0400 - f4e15d1: Merge branch 'master' of https://github.com/frankcabada/plt - Marissa
Ojeda
Thu Oct 20 16:55:38 2016 -0400 - 410541b: Scanner, parser, ast - Marissa Ojeda
Tue Oct 18 22:45:16 2016 -0400 - 23471d3: Ughhhh - frankcabada
Tue Oct 18 22:43:35 2016 -0400 - f40843b: This better work - frankcabada
Tue Oct 18 22:24:38 2016 -0400 - a4e26ff: we got this - frankcabada
Tue Oct 18 22:20:33 2016 -0400 - 86605f7: Trying again - frankcabada
Tue Oct 18 22:14:12 2016 -0400 - 90cb8ef: Added test comment to merge.sh - frankcabada
Tue Oct 18 22:08:06 2016 -0400 - 420bcdf: This should work now - frankcabada
Tue Oct 18 21:53:29 2016 -0400 - e7d3c9e: Test 6 this one fer sure is gona work... I hope - frankcabada
Tue Oct 18 21:52:17 2016 -0400 - 28c09e4: Test 5? - frankcabada
Tue Oct 18 21:50:06 2016 -0400 - 27954d8: Test 5? - frankcabada
Tue Oct 18 21:42:33 2016 -0400 - 18ed775: Test 4 - frankcabada
Tue Oct 18 21:39:25 2016 -0400 - 2b25b99: Test 3 - frankcabada
Tue Oct 18 21:36:44 2016 -0400 - a025676: Test 2 - frankcabada
Tue Oct 18 21:24:31 2016 -0400 - fbf1e7f: Testing travis-ci exit - frankcabada
Tue Oct 18 20:43:47 2016 -0400 - 7831235: yml test - frankcabada
Tue Oct 18 20:41:50 2016 -0400 - dc18053: Created .travis-ci.sh file and testing on hello world - frankcabada
```

```
Tue Oct 18 20:30:22 2016 -0400 - 07ccf9a: Added Daniel's environment variables to travis file - frankcabada
Tue Oct 18 20:29:25 2016 -0400 - 9049af2: Created .travis.yml file with my environment variables - frankcabada
Tue Oct 11 21:44:57 2016 -0400 - b451c40: started parser! - Michael Berkowitz
Tue Oct 11 21:21:31 2016 -0400 - 65fabfe: scanner.mll edited - Marissa Ojeda
Tue Oct 11 21:05:49 2016 -0400 - 780b523: scanner.mll - Marissa Ojeda
Tue Oct 11 20:34:27 2016 -0400 - 372bb1b: Created scanner.mll from example_scanner after removing two syntax erros
- frankcabada
Sun Oct 9 15:19:19 2016 -0400 - 1fe6a27: Added Marissa's example scanner - frankcabada
Sun Oct 9 15:08:52 2016 -0400 - 55ed27e: Initial commit - frankcabada
```

5. Translator Architecture

5.1 Diagram

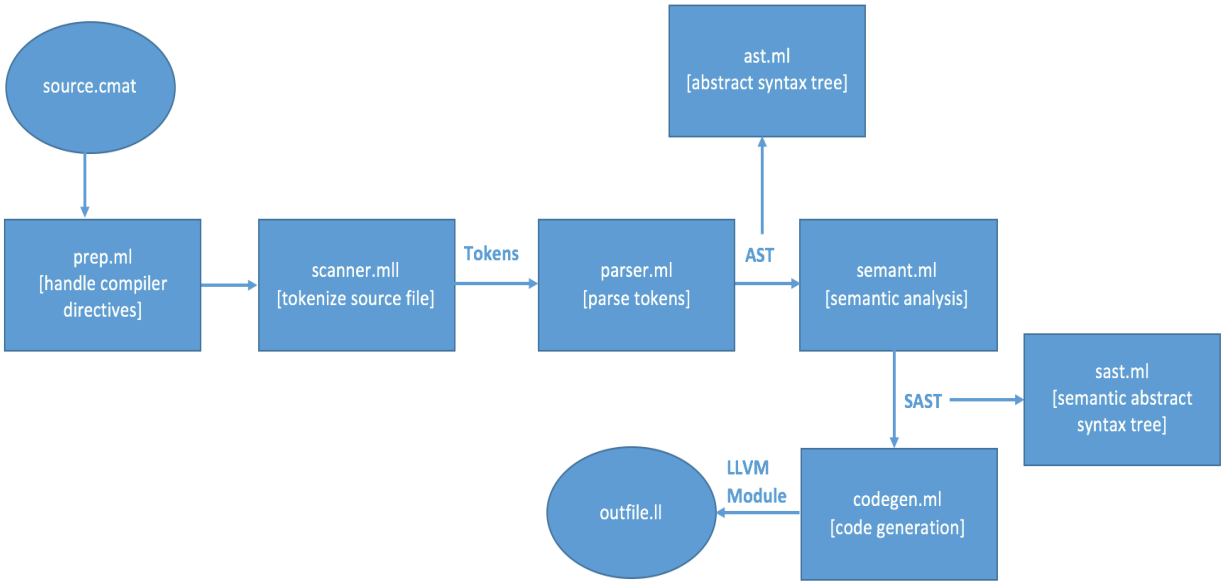


Figure 5.1 Compiler Representation

5.2 Compiler

The compiler used for CMAT is made up of the following modules (listed in order of use in compilation pipeline with a brief description of their functions):

- **prep.ml**: Scans the source file for compiler-level directives.
- **scanner.mll**: Converts a modified source file from the prep module into tokens.
- **parser.mly**: Takes in tokens from the scanner module, checks their syntactic correctness. If all of these checks are passed, it generates an abstract syntax tree (AST) representation of the program.
- **semant.ml**: Semantic checker that takes in an AST representation from the parser module and checks the semantic correctness. If all of these checks are passed, it generates a semantically-checked AST (SAST).
- **codegen.ml**: Takes in a semantically checked AST (SAST), assuming it is both syntactically and semantically valid, and translates each AST node into LLVM IR; it constructs expressions bottom-up and basic blocks for control-flow statements.

- **cmat.ml**: The top-level of the CMAT compiler; it invokes the scanner, parser, semant, and codegen modules to generate the LLVM IR, and dumps the module.

- **utils.ml:** Pretty printing our language.

And the following interfaces:

- **ast.ml**: A module which lays out the possible nodes in an AST representation of a syntactically correct CMAT program.

- **sast.ml**: A module which lays out the possible nodes in a SAST representation of a semantically correct CMAT program.

- **exceptions.ml**: Exceptions used in the language.

5.3 Prep

The prep module takes in a CMAT source file and scans it to find any include statements. If an include statement is found, the code from the file referenced is brought directly into the source, replacing the include directive.

5.4 Scanner

The scanner module takes in the modified source file from the prep module and converts it to a list of tokens. Comments are established here and removed from the intermediate representation of the program. The scanner also discards whitespace, tabs, and newlines are discarded.

5.5 Parser

The parser module scans through the tokens passed from the scanner and references the definitions set in parser.mly to construct an AST. The top-level node of the AST is a list of global variables and a list of function declarations. The parser outputs the following AST program representation in figure 5.2.

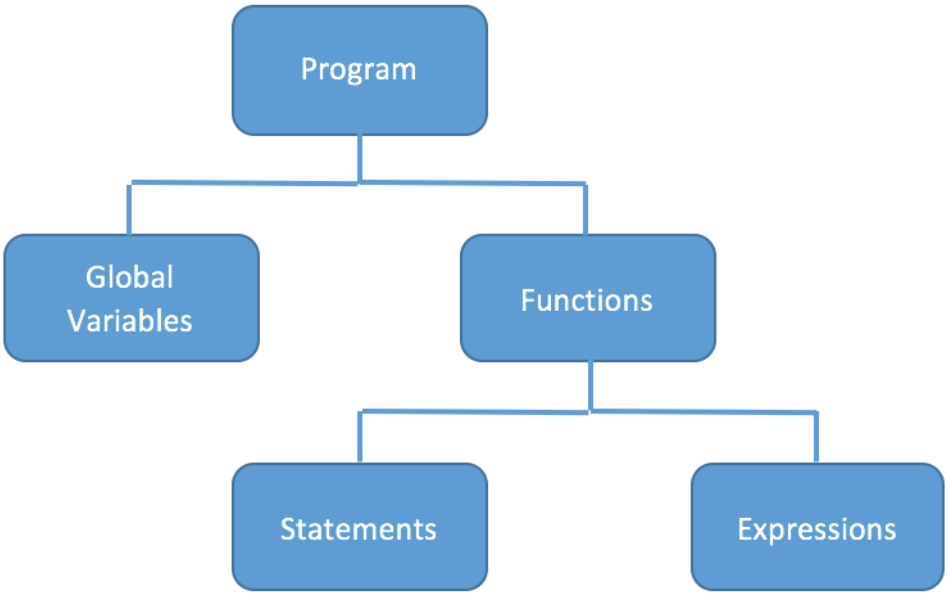


Figure 5.2 Abstract Syntax Tree Representation Program

5.6 Abstract Syntax Tree (AST)

The AST is a listed description of all possible nodes which a valid CMAT program syntax tree can contain.

5.7 Semantic Analyzer

The analyzer is called twice in the top-level of the compiler and performs substantial amount of error checking.

The first time it is called, the analyzer takes as input a list of global variables from the AST. Error checking on the global variables takes place such as ensuring that no variables are declared as void and that there are no duplicate declarations. The most important thing is that the global variables (if any) are added to a global symbol table which is then returned from this semantic check method.

The second time it is called, this global symbol table is passed in along with the list of the global variables and the list of the functions from the AST. The symbol table is passed in so the analyzer has access to the global variables and can perform error checking alongside the local and formal variables within functions. The first check of this second semantic runthrough is to make sure that all function names are distinct. Then the analyzer goes through each function and checks each expression and statement. The analyzer also ensures that non-void functions must return the correct type indicated by the function signature (including that functions with void return types do not return at all). Statements are checked to make sure they match the

semantic prototypes defined in the parser and ast. Some other semantic errors include ensuring that variable declarations occur at the top of functions, that declared variables hold the correct data type, that operations are performed on the correct data types, and matrix error checking. Additionally we create a table of reserved functions that cannot be declared (such as print_string, print_int, print_float, etc.) to check that no user-defined functions conflict with these names. Apart from error analysis, the analyzer constructs a SAST which is then used in code generation to produce LLVM IR.

5.8 Syntactically-Checked Abstract Syntax Tree (SAST)

The SAST is a listed description of all possible nodes which a valid CMAT program semantically-checked syntax tree can contain.

5.9 Code Generator

Code generation takes in the SAST produced from the semantic analyzer to construct the LLVM IR using the OCaml LLVM module, the AST and SAST modules, and the semantic analyzer module. The LLVM IR file contains assembly-like language that executes the operations indicated by the source program. The code generator parses the SAST and creates the LLVM IR corresponding to the structure of the indicated program. Finally, the code generator takes the LLVM IR and dumps it into an LLVM module which is written to an indicated output .ll file. This .ll file is a CMAT executable.

6. Test Plan and Scripts

6.1 Travis CI

As we decided to use Github for version control system, our group was advised by our TA to use Travis CI (Continuous Integration). Travis CI is a testing framework that integrates with Github to run a testing script every time someone pushes to a branch in the PLT repository. It runs the testing script on an essentially fresh image of Ubuntu 12.04.5 and reports whether the tests passed or failed. While our original plan was to use Travis CI to ensure that only commits that passed all tests would be allowed to merge into master, we found Travis to be a bit finicky in terms of automating Github merges. Thus, we settled with monitoring our own commit test feedback and determining on our own what was okay to push to master.

6.2 Scripts

In the top level of the PLT repository, there is a directory with the following structure:

```
/test/
|
|-- compiler/
    |
    |-- pass/
    |-- fail/
|-- parser/
    |
    |-- pass/
    |-- fail/
|-- scanner/
    |
    |-- pass/
    |-- fail/
```

Each directory contains a test.sh testing script which tests the resepective component of the compiler. Travis CI runs all of the test scripts and compares their output to a “golden standard” output file. The test scripts, all test input files, and intended output files can be found in §A.

6.2.1 Compiler

Compiler tests are complete tests of all parts of the translator architecture. Upon testing, all parts of the architecture - scanner, parser, AST, semantic analyzer, SAST, codegen, and the top level compiler - are compiled by ocamlbuild. The passing input test files, found in `test/compiler/pass`, are complete, valid, syntactically correct

programs which are run through all levels of the CMAT compiler. These programs print their output and this output is compared to a golden standard output file.

6.2.2 Parser

Parser tests are used to test that the parser correctly translates the tokens that come out of the scanner into an abstract syntax tree. Because the compiler tests ensure that all pieces of the architecture compile correctly, the parser tests simply use menhir with the `parser.mly` as an input. Menhir takes in a space-separated list of tokens and either accepts the input as syntactically valid and constructs the abstract syntax tree or rejects the input as syntactically incorrect. The sets of syntactically correct token strings are found in `test/parser/pass` while syntactically invalid token strings can be found in `test/parser/fail`.

6.2.3 Scanner

Scanner tests are focused on ensuring that the scanner correctly tokenizes input and rejects all words, characters, and strings which should not valid in CMAT. Tests which test valid input can be found in `test/scanner/pass`. Tests which test whether the scanner correctly rejects illegal characters and strings can be found in `test/scanner/fail`.

7. Conclusions

7.1 Mike

In retrospect, even with all of the warnings and advice from friends who had previously taken the class, I don't think I truly understood the scope of this project until about halfway through the semester. While I now have a thorough understanding of the basic components that comprise a compiler, I still have a lot to learn in terms of properly designing compilers and writing clear and concise code for them. The best part of this project, for me, was most certainly my team. We all were very understanding and reasonable with one another while communicating our goals and accomplishments and I felt that we all contributed equally to this behemoth of a project. The flexibility and dedication of my group went a long way toward making this project a much more enjoyable experience than it could have been. The worst part of this project for me personally was the stress of learning about compiler structure and OCaml for the first time while simultaneously trying to implement this new knowledge into an actual, working compiler. I think there were a lot of initial design decisions made without the foresight and understanding of a compiler design process that ended up restraining us toward the end of the project. The best representation of how I felt can be summarized by that gif of Grommit frantically laying down railroad tracks in front of him before his cart can run off the tracks (I'm sure you know the one). While there is a long list of things that I would do differently if given the chance to start over, I'm extremely proud of what we were able to accomplish in CMAT in terms of a first programming language.

7.2 Frank

There is no doubt in my mind that this has been the biggest and at times most confusing project that I have ever worked on. I'm glad that I was fortunate enough to have a dedicated team that worked well together because with their help I gained a better understanding of compilers throughout the semester each time we met to work on CMAT together. However, it took some time and a lot of work to fully grasp what was going on. Although we started working on this project early in the semester, I feel that a lot of time was spent trying to simply get something working. For this reason, we made some structural and design decisions that we now regret. We realized towards the end that certain features were much more difficult to implement due to these early choices and we either had to spend much time working around that or had to abandon a feature late in our project timeline. Therefore, it is not only important to start the project early but it is also important to establish a more flexible framework. Nonetheless, I am proud of what we implemented in CMAT and I appreciate all that I learned from working on it.

7.3 Daniel

This PLT project is definitely the most ambitious project I have been exposed to here at Columbia. The intricacies of building a language are far more complex than I ever imagined. Due to this, this challenge has definitely been a huge learning experience this semester. As I look back to the first project meeting, comparing what I knew about compilers then and now is a night and day difference. Our planning and communication definitely played a huge role in the outcome of our project. We tried our best to always stay on track and not fall behind despite many obstacles that we faced; and we faced many. With all this in mind, I am extremely proud of what we were able to accomplish together and very excited knowing that CMAT is by my side when I need to tackle matrix problems.

7.4 Marissa

Looking back, I am very glad we started early and established 3-4 meeting times a week. We realized very early that it was best to split up into pairs to accomplish tasks. For example, one pair set up Travis CI and the testing framework and another pair worked on the scanner and parser. As system architect, I purposely tried to work on every aspect of the compiler. This was a great way to learn the different parts of the compiler. As a result, I had a cohesive understanding of our compiler. As I was working on the semantic analyzer, I grew an appreciation for ocaml. I much prefer different functions defined rather than one main function as it was done for MicroC semantic analyzer and code gen. In the end, this project was manageable because of my team's work ethic. We were all very committed to make sure we didn't pull all-nighters toward the due date.

A. Full Code Listing

A.1 plt/hello_world

A.1.1 plt/hello_world/scripts/build.sh

```
#!/bin/bash

cp ../src/scanner.mll ./scanner.mll
cp ../src/parser.mly ./parser.mly
cp ../src/ast.ml ./ast.ml
cp ../src/sast.ml ./sast.ml
cp ../src/semant.ml ./semant.ml
cp ../src/exceptions.ml ./exceptions.ml
cp ../src/utils.ml ./utils.ml
cp ../src/codegen.ml ./codegen.ml
cp ../src/prep.ml ./prep.ml
cp ../src/cmat.ml ./cmat.ml

ocamlbuild -j 0 -r -use-ocamlfind -pkgs
str,llvm,llvm.analysis,llvm.bitwriter,llvm.bitreader,llvm.linker,llvm.target cmat.native
```

A.1.2 plt/hello_world/scripts/clean.sh

```
#!/bin/bash

rm -rf _build *.cmo *.cmi *.res *.ml* cmat.native build.log hello_world.ll
```

A.1.3 plt/hello_world/scripts/test.sh

```
#!/bin/bash

./cmat.native -c hello_world.cmat hello_world.ll
lli hello_world.ll > hello_world.res
diff hello_world.res hello_world.out > /dev/null
```

A.1.4 plt/hello_world/hello_world.cmat

```
int main() {
  print_string("Hello world!");
  return 0;
}
```

A.1.5 plt/hello_world/hello_world.out

```
Hello world!
```

A.2 plt/scripts

A.2.1 plt/scripts/build.sh

```
#!/bin/bash

cp ./src/scanner.mll ./scanner.mll
cp ./src/parser.mly ./parser.mly
cp ./src/ast.ml ./ast.ml
cp ./src/sast.ml ./sast.ml
cp ./src/semant.ml ./semant.ml
cp ./src/exceptions.ml ./exceptions.ml
cp ./src/utils.ml ./utils.ml
cp ./src/codegen.ml ./codegen.ml
cp ./src/prep.ml ./prep.ml
cp ./src/cmat.ml ./cmat.ml

ocamlbuild -j 0 -r -use-ocamlfind -pkgs
str,llvm,llvm.analysis,llvm.bitwriter,llvm.bitreader,llvm.linker,llvm.target cmat.native
```

A.2.2 plt/scripts/clean.sh

```
#!/bin/bash

rm -rf _build build.log cmat.native *.cmo *.cmi *.ml* *.ll
```

A.2.3 plt/scripts/test.sh

```
#! /bin/bash

cd ./test/scanner
./scripts/build.sh
./scripts/test.sh
./scripts/clean.sh
cd ../parser
./scripts/test.sh
./scripts/clean.sh
cd ../compiler
./scripts/build.sh > build.log
./scripts/test.sh
./scripts/clean.sh
cd ../../hello_world
./scripts/build.sh > build.log
./cmat.native -c hello_world.cmat hello_world.ll
lli hello_world.ll > hello_world.res
diff -q hello_world.out hello_world.res
./scripts/clean.sh
```

A.3 plt/src

A.3.1 plt/src/ast.ml

```
(*
 * COMS4115: CMAT AST
 *
 * Authors:
 *  - Marissa Ojeda
 *  - Daniel Rojas
 *  - Mike Berkowitz
 *  - Frank Cabada
 *)

(* Binary Operators *)
type op = Add | Sub | Mult | Div | Equal | Neq | Less | Leq | Greater | Geq | And | Or

(* Unary Operators *)
type uop = Neg | Not | Inc | Dec

(* Nums *)
type num = Int_lit of int | Float_lit of float

(* Types *)
type primitives = Int    | Bool    | Void    | String    | Float
    | Vector of primitives * num
    | Matrix of primitives * num * num

type datatype = Datatype of primitives

type formal = Formal of datatype*string
type local = Local of datatype*string
type var_dec = datatype*string

(* Expressions *)
type expr =
    | Num_lit of num
    | Bool_lit of bool
    | String_lit of string
    | Matrix_lit of num list list
    | Vector_lit of num list
    | Id of string
    | Noexpr
    | Null
    | Binop of expr * op * expr
    | Unop of uop * expr
    | Assign of expr * expr
    | Call of string * expr list
    | Vector_access of string * expr
    | Matrix_access of string * expr * expr
    | Matrix_row of string * expr
    | Matrix_col of string * expr
```

```
    | Rows of string
    | Cols of string
    | Len of string
    | Transpose of string

(* Statements *)
type stmt =
    | Block of stmt list
    | Expr of expr
    | If of expr * stmt * stmt
    | For of expr * expr * expr * stmt
    | While of expr * stmt
    | Return of expr

(* Function Declarations *)
type func_decl = {
    return_type : datatype;
    fname       : string;
    formals     : formal list;
    locals      : local list;
    body        : stmt list;
}

(* Start Symbol *)
type program = var_dec list * func_decl list
```

A.3.2 plt/src/cmat.ml

```
(*
 * COMS4115: CMAT Top-Level Compiler
 *
 * Authors:
 *  - Marissa Ojeda
 *  - Daniel Rojas
 *  - Mike Berkowitz
 *  - Frank Cabada
 *)

open Scanner
open Parser
open Ast
open Utils
open Codegen
open Semant
open Prep
open Llvm

type action = Ast | LLVM_IR | Compile

let _ =
       let action = if Array.length Sys.argv > 1
       then List.assoc Sys.argv.(1)
              [ ("-a",Ast); ("-l",LLVM_IR); ("-c",Compile) ]
```

```
        else Compile in
    let infile =
    if (Array.length Sys.argv > 2)
        then Sys.argv.(2)
        else raise(Exceptions.NoFileArgument)
    in
    let outfile =
    if (Array.length Sys.argv > 3 && action=Compile)
        then Sys.argv.(3)
        else "out.ll" in
    let strin = Prep.process infile in
    let lexbuf = Lexing.from_string strin in
    let ast = Parser.program Scanner.token lexbuf in

    let sast =
    let gst = Semant.check_var_decls (fst ast) in
    Semant.check_functions gst (fst ast) (snd ast) in

    match action with
    Ast -> print_string(Utils.string_of_program ast)
    | LLVM_IR -> print_string(Llvm.string_of_llmodule(Codegen.translate sast))
    | Compile -> let m = Codegen.translate sast in
        Llvm_analysis.assert_valid_module m; (* Useful built-in check *)
        print_module (outfile) (m);
```

A.3.3 plt/src/codegen.ml

```
(*
 * COMS4115: CMAT Code Generator
 *
 * Authors:
 *  - Marissa Ojeda
 *  - Daniel Rojas
 *  - Mike Berkowitz
 *  - Frank Cabada
 *)

open Llvm
open Ast
open Sast
open Semant
module L = Llvm
module A = Ast
module S = Sast

module StringMap = Map.Make(String)

let translate(globals, functions) =

    let context = L.global_context() in
    let the_module = L.create_module context "CMAT"

    and i32_t     = L.i32_type context
    and i1_t      = L.i1_type context
```

```
    and i8_t      = L.i8_type context
    and float_t   = L.double_type context
    and void_t    = L.void_type context
    and array_t   = L.array_type
    and pointer_t = L.pointer_type
    in

    let ltype_of_typ = function
          A.Int     -> i32_t
        | A.Float   -> float_t
        | A.Bool    -> i1_t
        | A.Void    -> void_t
        | A.String  -> pointer_t i8_t
        | A.Vector(typ, size) ->
            let size' = match size with Int_lit(s) -> s | _ ->
raise(Exceptions.InvalidVectorDimension) in
            (match typ with
                A.Int      ->  array_t i32_t size'
                | A.Float  -> array_t float_t size'
                | _ -> raise(Exceptions.UnsupportedVectorType))
        | A.Matrix(typ, rows, cols) ->
            let rows' = match rows with Int_lit(s) -> s | _ ->
raise(Exceptions.InvalidMatrixDimension) in
            let cols' = match cols with Int_lit(s) -> s | _ ->
raise(Exceptions.InvalidMatrixDimension) in
            (match typ with
                A.Int      -> array_t (array_t i32_t cols') rows'
                | A.Float  -> array_t (array_t float_t cols') rows'
                | _ -> raise(Exceptions.UnsupportedMatrixType))
    in

    let ltype_of_datatype = function
        A.Datatype(p) -> ltype_of_typ p
    in

    let global_vars =
        let global_var m (t,n) =
            let init = L.const_int (ltype_of_datatype 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.var_arg_function_type i32_t [| L.pointer_type i8_t |]
    in
    let printf_func =
        L.declare_function "printf" printf_t the_module
    in

    let function_decls =
        let function_decl m fdecl =
            let name = fdecl.S.sfname
            and formal_types = Array.of_list
                (List.map (function A.Formal(t,s) -> ltype_of_datatype t) fdecl.S.sformals) in
```

```
            let ftype =
                L.function_type (ltype_of_datatype fdecl.S.sreturn_type) 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

    let build_function_body fdecl =

        let (the_function, _) = StringMap.find fdecl.S.sfname function_decls in

        (* Create an instruction builder *)
        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 float_format_str = L.build_global_stringptr "%f\t" "fmt" builder
        and string_format_str = L.build_global_stringptr "%s\n" "fmt" builder
        in

        let local_vars =
            let add_formal m (t, n) p = L.set_value_name n p;
            let local = L.build_alloca (ltype_of_datatype t) n builder in
            ignore (L.build_store p local builder);
            StringMap.add n local m
        in

        let add_local m (t, n) =
            let local_var = L.build_alloca (ltype_of_datatype t) n builder
            in StringMap.add n local_var m
        in

        let formals = List.fold_left2 add_formal StringMap.empty
            (List.map (function A.Formal(t,n) -> (t,n)) fdecl.S.sformals) (Array.to_list (L.params
the_function)) in
            List.fold_left add_local formals (List.map (function A.Local(t,n) -> (t,n))
fdecl.S.slocals)
        in

        let lookup n = try StringMap.find n local_vars
            with Not_found -> StringMap.find n global_vars
        in

        let build_matrix_access access_i access_j s i1 i2 i3 builder isAssign =
            let rows = L.array_length (L.type_of (L.build_load (L.build_gep (lookup s) [| L.const_int
i32_t 0 |] s builder) s builder)) in
            let cols = L.array_length (L.type_of (L.build_load (L.build_gep (lookup s) [| L.const_int
i32_t 0; L.const_int i32_t 0 |] s builder) s builder)) in
            if ((rows < access_i) && (cols > access_j)) then
raise(Exceptions.MatrixOutOfBoundsAccess(""));
            if isAssign
                then L.build_gep (lookup s) [| i1; i2; i3 |] s builder
                else L.build_load (L.build_gep (lookup s) [| i1; i2; i3 |] s builder) s builder
        in

        let build_vector_access access_i s i1 i2 builder isAssign =
```

```
            let len = L.array_length (L.type_of (L.build_load (L.build_gep (lookup s) [| L.const_int
i32_t 0 |] s builder) s builder)) in
            if (len < access_i) then raise(Exceptions.VectorOutOfBoundsAccess(""));
            if isAssign
                then L.build_gep (lookup s) [| i1; i2 |] s builder
                else L.build_load (L.build_gep (lookup s) [| i1; i2 |] s builder) s builder
        in

        let rec expr builder = function
            S.SNum_lit(SInt_lit(n))     -> L.const_int i32_t n
            | S.SNum_lit(SFloat_lit(f)) -> L.const_float float_t f
            | S.SBool_lit b             -> L.const_int i1_t (if b then 1 else 0)
            | S.SString_lit s           -> L.build_global_stringptr s "tmp" builder
            | S.SNoexpr                 -> L.const_int i32_t 0
            | S.SId (s, d)              -> L.build_load (lookup s) s builder
            | S.SAssign (se1, se2, d)   ->
                let se1' =
                    (match se1 with
                        S.SId(s,_) -> (lookup s)
                        | S.SMatrix_access(s, i1, j1, d) ->
                            let i = expr builder i1 and j = expr builder j1 in
                            let access_i = (match i1 with S.SNum_lit(SInt_lit(n)) -> n | _ -> -1)
                            and access_j = (match j1 with S.SNum_lit(SInt_lit(n)) -> n | _ -> -1) in
                                build_matrix_access access_i access_j s (L.const_int i32_t 0) i j
builder true
                        | S.SVector_access(s, i1, d) ->
                            let i = expr builder i1 in
                            let access_i = (match i1 with S.SNum_lit(SInt_lit(n)) -> n | _ -> -1) in
                                build_vector_access access_i s (L.const_int i32_t 0) i builder true
                        | _ -> raise(Exceptions.AssignLHSMustBeAssignable))
                and se2' = expr builder se2 in
                ignore (L.build_store se2' se1' builder); se2'
            | S.SBinop (e1, op, e2, d)  ->
                let type1 = Semant.get_type_from_sexpr e1 in
                let type2 = Semant.get_type_from_sexpr e2 in
                let e1' = expr builder e1
                and e2' = expr builder e2 in

                let int_bops op e1' e2' =
                    match op with
                        A.Add     -> L.build_add e1' e2' "tmp" builder
                        | A.Sub     -> L.build_sub e1' e2' "tmp" builder
                        | A.Mult    -> L.build_mul e1' e2' "tmp" builder
                        | A.Div     -> L.build_sdiv e1' e2' "tmp" builder
                        | A.Equal   -> L.build_icmp L.Icmp.Eq e1' e2' "tmp" builder
                        | A.Neq     -> L.build_icmp L.Icmp.Ne e1' e2' "tmp" builder
                        | A.Less    -> L.build_icmp L.Icmp.Slt e1' e2' "tmp" builder
                        | A.Leq     -> L.build_icmp L.Icmp.Sle e1' e2' "tmp" builder
                        | A.Greater -> L.build_icmp L.Icmp.Sgt e1' e2' "tmp" builder
                        | A.Geq     -> L.build_icmp L.Icmp.Sge e1' e2' "tmp" builder
                        | A.And     -> L.build_and e1' e2' "tmp" builder
                        | A.Or      -> L.build_or e1' e2' "tmp" builder
                in
```

```
            let float_bops op e1' e2' =
                match op with
                    A.Add       -> L.build_fadd e1' e2' "tmp" builder
                  | A.Sub       -> L.build_fsub e1' e2' "tmp" builder
                  | A.Mult      -> L.build_fmul e1' e2' "tmp" builder
                  | A.Div       -> L.build_fdiv e1' e2' "tmp" builder
                  | A.Equal     -> L.build_fcmp L.Fcmp.Oeq e1' e2' "tmp" builder
                  | A.Neq       -> L.build_fcmp L.Fcmp.One e1' e2' "tmp" builder
                  | A.Less      -> L.build_fcmp L.Fcmp.Olt e1' e2' "tmp" builder
                  | A.Leq       -> L.build_fcmp L.Fcmp.Ole e1' e2' "tmp" builder
                  | A.Greater   -> L.build_fcmp L.Fcmp.Ogt e1' e2' "tmp" builder
                  | A.Geq       -> L.build_fcmp L.Fcmp.Oge e1' e2' "tmp" builder
                  | _           -> raise(Exceptions.IllegalFloatBinop)
            in

            let bool_bops op e1' e2' =
                match op with
                    | A.And   -> L.build_and e1' e2' "tmp" builder
                    | A.Or    -> L.build_or e1' e2' "tmp" builder
                    | _       -> raise(Exceptions.IllegalBoolBinop)
            in

            let vector_bops iorf n_i n op e1 e2 =
                let lhs_str = (match e1 with SId(s,_) -> s | _ -> "") in
                let rhs_str = (match e2 with SId(s,_) -> s | _ -> "") in
                match iorf with
                    "int" ->
                        (match op with
                            A.Add       ->
                                let tmp_v = L.build_alloca (array_t i32_t n_i) "tmpvec" builder
in
                                for i=0 to n_i do
                                        let v1 = build_vector_access n_i lhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) builder false in
                                        let v2 = build_vector_access n_i rhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) builder false in
                                        let add_res = L.build_add v1 v2 "tmp" builder in
                                        let ld = L.build_gep tmp_v [| L.const_int i32_t 0;
L.const_int i32_t i |] "tmpvec" builder in
                                        ignore(build_store add_res ld builder);
                                done;
                                L.build_load (L.build_gep tmp_v [| L.const_int i32_t 0 |]
"tmpvec" builder) "tmpvec" builder
                            | A.Sub     ->
                                let tmp_v = L.build_alloca (array_t i32_t n_i) "tmpvec" builder
in
                                for i=0 to n_i do
                                        let v1 = build_vector_access n_i lhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) builder false in
                                        let v2 = build_vector_access n_i rhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) builder false in
                                        let add_res = L.build_sub v1 v2 "tmp" builder in
                                        let ld = L.build_gep tmp_v [| L.const_int i32_t 0;
L.const_int i32_t i |] "tmpvec" builder in
```

```
                                        ignore(build_store add_res ld builder);
                                    done;
                                    L.build_load (L.build_gep tmp_v [| L.const_int i32_t 0 |]
"tmpvec" builder) "tmpvec" builder
                                | A.Mult     ->
                                    let first_typ = Semant.get_type_from_sexpr e1 in
                                    let tmp_v = L.build_alloca (array_t i32_t n_i) "tmpvec" builder
in
                                    (match first_typ with
                                        Datatype(Int) ->
                                            for i=0 to n_i do
                                                let v2 = build_vector_access n_i rhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) builder false in
                                                let add_res = L.build_mul (build_load (lookup
lhs_str) "tmp" builder) v2 "tmp" builder in
                                                let ld = L.build_gep tmp_v [| L.const_int i32_t 0;
L.const_int i32_t i |] "tmpvec" builder in
                                                ignore(build_store add_res ld builder);
                                            done;
                                            L.build_load (L.build_gep tmp_v [| L.const_int i32_t 0 |]
"tmpvec" builder) "tmpvec" builder
                                        | Datatype(Matrix(Int,r1,c1)) ->
                                            let r1_i = (match r1 with Int_lit(n) -> n | _ -> -1) in
                                            let c1_i = (match c1 with Int_lit(n) -> n | _ -> -1) in
                                            let tmp_s = L.build_alloca i32_t "tmpsum" builder in
                                            let tmp_v = L.build_alloca (array_t i32_t r1_i) "tmpvec"
builder in
                                            ignore(L.build_store (L.const_int i32_t 0) tmp_s
builder);
                                            for i=0 to (r1_i-1) do
                                                ignore(L.build_store (L.const_int i32_t 0) tmp_s
builder);
                                                for j=0 to (c1_i-1) do
                                                        let m1 = build_matrix_access i j lhs_str
(L.const_int i32_t 0) (L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                                        let v2 = build_vector_access j rhs_str
(L.const_int i32_t 0) (L.const_int i32_t j) builder false in
                                                        let mult_res = L.build_mul m1 v2 "tmp"
builder in
                                                        ignore(L.build_store (L.build_add mult_res
(L.build_load tmp_s "addtmp" builder) "tmp" builder) tmp_s builder);
                                                done;
                                                let ld = L.build_gep tmp_v [| L.const_int i32_t 0;
L.const_int i32_t i |] "tmpvec" builder in
                                                ignore(build_store (L.build_load tmp_s "restmp"
builder) ld builder);
                                            done;
                                            L.build_load (L.build_gep tmp_v [| L.const_int i32_t 0 |]
"tmpvec" builder) "tmpvec" builder
                                        | _ -> L.const_int i32_t 0)
                                | _         -> raise(Exceptions.IllegalVectorBinop))
                        | "float" ->
                            (match op with
                                A.Add       ->
```

```
                                    let lhs_str = (match e1 with SId(s,_) -> s | _ -> "") in
                                    let rhs_str = (match e2 with SId(s,_) -> s | _ -> "") in
                                    let tmp_v = L.build_alloca (array_t float_t n_i) "tmpvec" builder
in
                                    for i=0 to n_i do
                                            let v1 = build_vector_access n_i lhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) builder false in
                                            let v2 = build_vector_access n_i rhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) builder false in
                                            let add_res = L.build_fadd v1 v2 "tmp" builder in
                                            let ld = L.build_gep tmp_v [| L.const_int i32_t 0;
L.const_int i32_t i |] "tmpvec" builder in
                                            ignore(build_store add_res ld builder);
                                    done;
                                    L.build_load (L.build_gep tmp_v [| L.const_int i32_t 0 |]
"tmpvec" builder) "tmpvec" builder
                                | A.Sub     ->
                                    let lhs_str = (match e1 with SId(s,_) -> s | _ -> "") in
                                    let rhs_str = (match e2 with SId(s,_) -> s | _ -> "") in
                                    let tmp_v = L.build_alloca (array_t float_t n_i) "tmpvec" builder
in
                                    for i=0 to n_i do
                                            let v1 = build_vector_access n_i lhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) builder false in
                                            let v2 = build_vector_access n_i rhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) builder false in
                                            let add_res = L.build_fsub v1 v2 "tmp" builder in
                                            let ld = L.build_gep tmp_v [| L.const_int i32_t 0;
L.const_int i32_t i |] "tmpvec" builder in
                                            ignore(build_store add_res ld builder);
                                    done;
                                    L.build_load (L.build_gep tmp_v [| L.const_int i32_t 0 |]
"tmpvec" builder) "tmpvec" builder
                                | A.Mult    ->
                                    let first_typ = Semant.get_type_from_sexpr e1 in
                                    let tmp_v = L.build_alloca (array_t float_t n_i) "tmpvec" builder
in
                                    (match first_typ with
                                        Datatype(Float) ->
                                            for i=0 to n_i do
                                                let v2 = build_vector_access n_i rhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) builder false in
                                                let add_res = L.build_fmul (build_load (lookup
lhs_str) "tmp" builder) v2 "tmp" builder in
                                                let ld = L.build_gep tmp_v [| L.const_int i32_t 0;
L.const_int i32_t i |] "tmpvec" builder in
                                                ignore(build_store add_res ld builder);
                                            done;
                                            L.build_load (L.build_gep tmp_v [| L.const_int i32_t 0 |]
"tmpvec" builder) "tmpvec" builder
                                        | Datatype(Matrix(Float,r1,c1)) ->
                                            let r1_i = (match r1 with Int_lit(n) -> n | _ -> -1) in
                                            let c1_i = (match c1 with Int_lit(n) -> n | _ -> -1) in
                                            let tmp_s = L.build_alloca float_t "tmpsum" builder in
```

```
                                        let tmp_v = L.build_alloca (array_t float_t r1_i)
"tmpvec" builder in
                                        ignore(L.build_store (L.const_float float_t 0.0) tmp_s
builder);
                                        for i=0 to (r1_i-1) do
                                            ignore(L.build_store (L.const_float float_t 0.0)
tmp_s builder);
                                            for j=0 to (c1_i-1) do
                                                let m1 = build_matrix_access i j lhs_str
(L.const_int i32_t 0) (L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                                let v2 = build_vector_access j rhs_str
(L.const_int i32_t 0) (L.const_int i32_t j) builder false in
                                                let mult_res = L.build_fmul m1 v2 "tmp"
builder in
                                                ignore(L.build_store (L.build_fadd mult_res
(L.build_load tmp_s "addtmp" builder) "tmp" builder) tmp_s builder);
                                            done;
                                            let ld = L.build_gep tmp_v [| L.const_int i32_t 0;
L.const_int i32_t i |] "tmpvec" builder in
                                            ignore(build_store (L.build_load tmp_s "restmp"
builder) ld builder);
                                        done;
                                        L.build_load (L.build_gep tmp_v [| L.const_int i32_t 0
|] "tmpvec" builder) "tmpvec" builder
                                    | _ -> L.const_int i32_t 0)
                                | _         -> raise(Exceptions.IllegalVectorBinop))
                        | _ -> L.const_int i32_t 0
                in

                let matrix_bops iorf r_i c_i r c op e1 e2 =
                    let lhs_str = (match e1 with SId(s,_) -> s | _ -> "") in
                    let rhs_str = (match e2 with SId(s,_) -> s | _ -> "") in
                    match iorf with
                        "int" ->
                            (match op with
                                A.Add       ->
                                    let tmp_m = L.build_alloca (array_t (array_t i32_t c_i) r_i)
"tmpmat" builder in
                                    for i=0 to (r_i-1) do
                                        for j=0 to (c_i-1) do
                                            let m1 = build_matrix_access r_i c_i lhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                            let m2 = build_matrix_access r_i c_i rhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                            let add_res = L.build_add m1 m2 "tmp" builder in
                                            let ld = L.build_gep tmp_m [| L.const_int i32_t 0;
L.const_int i32_t i; L.const_int i32_t j |] "tmpmat" builder in
                                            ignore(build_store add_res ld builder);
                                        done
                                    done;
                                    L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |]
"tmpmat" builder) "tmpmat" builder
                                | A.Sub     ->
                                    let tmp_m = L.build_alloca (array_t (array_t i32_t c_i) r_i)
```

```
"tmpmat" builder in
                                    for i=0 to (r_i-1) do
                                        for j=0 to (c_i-1) do
                                            let m1 = build_matrix_access r_i c_i lhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                            let m2 = build_matrix_access r_i c_i rhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                            let add_res = L.build_sub m1 m2 "tmp" builder in
                                            let ld = L.build_gep tmp_m [| L.const_int i32_t 0;
L.const_int i32_t i; L.const_int i32_t j |] "tmpmat" builder in
                                            ignore(build_store add_res ld builder);
                                        done
                                    done;
                                    L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |]
"tmpmat" builder) "tmpmat" builder
                                | A.Mult    ->
                                    let first_typ = Semant.get_type_from_sexpr e1 in
                                    let tmp_m = L.build_alloca (array_t (array_t i32_t c_i) r_i)
"tmpmat" builder in
                                    (match first_typ with
                                        Datatype(Int) ->
                                            for i=0 to (r_i-1) do
                                                for j=0 to (c_i-1) do
                                                    let m2 = build_matrix_access r_i c_i rhs_str
(L.const_int i32_t 0) (L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                                    let add_res = L.build_mul (build_load (lookup
lhs_str) "tmp" builder) m2 "tmp" builder in
                                                    let ld = L.build_gep tmp_m [| L.const_int i32_t
0; L.const_int i32_t i; L.const_int i32_t j |] "tmpmat" builder in
                                                    ignore(build_store add_res ld builder);
                                                done
                                            done;
                                            L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |]
"tmpmat" builder) "tmpmat" builder
                                        | Datatype(Matrix(Int,r1,c1)) ->
                                            let tmp_s = L.build_alloca i32_t "tmpsum" builder in
                                            let c1_i = (match c1 with Int_lit(n) -> n | _ -> -1) in
                                            ignore(L.build_store (L.const_int i32_t 0) tmp_s
builder);
                                            for i=0 to (r_i-1) do
                                                for j=0 to (c_i-1) do
                                                    ignore(L.build_store (L.const_int i32_t 0) tmp_s
builder);
                                                    for k=0 to (c1_i-1) do
                                                        let m1 = build_matrix_access r_i c1_i lhs_str
(L.const_int i32_t 0) (L.const_int i32_t i) (L.const_int i32_t k) builder false in
                                                        let m2 = build_matrix_access c1_i c_i rhs_str
(L.const_int i32_t 0) (L.const_int i32_t k) (L.const_int i32_t j) builder false in
                                                        let mult_res = L.build_mul m1 m2 "tmp"
builder in
                                                        ignore(L.build_store (L.build_add mult_res
(L.build_load tmp_s "addtmp" builder) "tmp" builder) tmp_s builder);
                                                    done;
                                                    let ld = L.build_gep tmp_m [| L.const_int i32_t
```

```
0; L.const_int i32_t i; L.const_int i32_t j |] "tmpmat" builder in
                                                    ignore(build_store (L.build_load tmp_s "restmp"
builder) ld builder);
                                                done
                                            done;
                                            L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |]
"tmpmat" builder) "tmpmat" builder
                                        | _ -> L.const_int i32_t 0)
                                | _         -> raise(Exceptions.IllegalMatrixBinop))
                            | "float" ->
                                (match op with
                                    A.Add       ->
                                        let tmp_m = L.build_alloca (array_t (array_t float_t c_i) r_i)
"tmpmat" builder in
                                        for i=0 to (r_i-1) do
                                            for j=0 to (c_i-1) do
                                                let m1 = build_matrix_access r_i c_i lhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                                let m2 = build_matrix_access r_i c_i rhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                                let add_res = L.build_fadd m1 m2 "tmp" builder in
                                                let ld = L.build_gep tmp_m [| L.const_int i32_t 0;
L.const_int i32_t i; L.const_int i32_t j |] "tmpmat" builder in
                                                ignore(build_store add_res ld builder);
                                            done
                                        done;
                                        L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |]
"tmpmat" builder) "tmpmat" builder
                                    | A.Sub     ->
                                        let tmp_m = L.build_alloca (array_t (array_t float_t c_i) r_i)
"tmpmat" builder in
                                        for i=0 to (r_i-1) do
                                            for j=0 to (c_i-1) do
                                                let m1 = build_matrix_access r_i c_i lhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                                let m2 = build_matrix_access r_i c_i rhs_str (L.const_int
i32_t 0) (L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                                let add_res = L.build_fsub m1 m2 "tmp" builder in
                                                let ld = L.build_gep tmp_m [| L.const_int i32_t 0;
L.const_int i32_t i; L.const_int i32_t j |] "tmpmat" builder in
                                                ignore(build_store add_res ld builder);
                                            done
                                        done;
                                        L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |]
"tmpmat" builder) "tmpmat" builder
                                    | A.Mult    ->
                                        let first_typ = Semant.get_type_from_sexpr e1 in
                                        let tmp_m = L.build_alloca (array_t (array_t float_t c_i) r_i)
"tmpmat" builder in
                                        (match first_typ with
                                            Datatype(Float) ->
                                                for i=0 to (r_i-1) do
                                                    for j=0 to (c_i-1) do
                                                        let m2 = build_matrix_access r_i c_i rhs_str
```

```
(L.const_int i32_t 0) (L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                                    let add_res = L.build_fmul (build_load (lookup
lhs_str) "tmp" builder) m2 "tmp" builder in
                                                    let ld = L.build_gep tmp_m [| L.const_int i32_t
0; L.const_int i32_t i; L.const_int i32_t j |] "tmpmat" builder in
                                                    ignore(build_store add_res ld builder);
                                                done
                                            done;
                                            L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |]
"tmpmat" builder) "tmpmat" builder
                                        | Datatype(Matrix(Float,r1,c1)) ->
                                            let tmp_s = L.build_alloca float_t "tmpsum" builder in
                                            let c1_i = (match c1 with Int_lit(n) -> n | _ -> -1) in
                                            ignore(L.build_store (L.const_float float_t 0.0) tmp_s
builder);
                                            for i=0 to (r_i-1) do
                                                for j=0 to (c_i-1) do
                                                    ignore(L.build_store (L.const_float float_t 0.0)
tmp_s builder);
                                                    for k=0 to (c1_i-1) do
                                                        let m1 = build_matrix_access r_i c1_i lhs_str
(L.const_int i32_t 0) (L.const_int i32_t i) (L.const_int i32_t k) builder false in
                                                        let m2 = build_matrix_access c1_i c_i rhs_str
(L.const_int i32_t 0) (L.const_int i32_t k) (L.const_int i32_t j) builder false in
                                                        let mult_res = L.build_fmul m1 m2 "tmp"
builder in
                                                        ignore(L.build_store (L.build_fadd mult_res
(L.build_load tmp_s "addtmp" builder) "tmp" builder) tmp_s builder);
                                                    done;
                                                    let ld = L.build_gep tmp_m [| L.const_int i32_t
0; L.const_int i32_t i; L.const_int i32_t j |] "tmpmat" builder in
                                                    ignore(build_store (L.build_load tmp_s "restmp"
builder) ld builder);
                                                done
                                            done;
                                            L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |]
"tmpmat" builder) "tmpmat" builder
                                        | _ -> L.const_int i32_t 0)
                                | _         -> raise(Exceptions.IllegalMatrixBinop))
                        | _ -> L.const_int i32_t 0
                in

                let cast lhs rhs lhsType rhsType =
                    match (lhsType, rhsType) with
                        (Datatype(Int), Datatype(Int))          -> (lhs, rhs), Datatype(Int)
                        | (Datatype(Float), Datatype(Float))    ->  (lhs, rhs), Datatype(Float)
                        | (Datatype(Bool), Datatype(Bool))      ->  (lhs, rhs), Datatype(Bool)
                        | (Datatype(Int), Datatype(Float))      ->   (build_sitofp lhs float_t "tmp"
builder, rhs), Datatype(Float)
                        | (Datatype(Float), Datatype(Int))      ->   (lhs, build_sitofp rhs float_t
"tmp" builder), Datatype(Float)
                        | (Datatype(Int), Datatype(Bool))       -> (lhs, rhs), Datatype(Bool)
                        | (Datatype(Bool), Datatype(Int))       -> (lhs, rhs), Datatype(Int)
                        | (Datatype(Int), Datatype(Vector(Int, n1))) ->
```

```
                        (lhs, rhs), Datatype(Vector(Int, n1))
                    | (Datatype(Float), Datatype(Vector(Float, n1))) ->
                        (lhs, rhs), Datatype(Vector(Float, n1))
                    | (Datatype(Vector(Int, n1)), Datatype(Vector(Int, n2))) ->
                        (lhs, rhs), Datatype(Vector(Int, n1))
                    | (Datatype(Vector(Float, n1)), Datatype(Vector(Float, n2))) ->
                        (lhs, rhs), Datatype(Vector(Float, n1))
                    | (Datatype(Int), Datatype(Matrix(Int,r1,c2))) ->
                        (lhs, rhs), Datatype(Matrix(Int, r1, c2))
                    | (Datatype(Float), Datatype(Matrix(Float,r1,c2))) ->
                        (lhs, rhs), Datatype(Matrix(Float, r1, c2))
                    | (Datatype(Matrix(Int,r1,c1)), Datatype(Vector(Int,n))) ->
                         (lhs, rhs), Datatype(Vector(Int, r1))
                    | (Datatype(Matrix(Float,r1,c1)), Datatype(Vector(Float,n))) ->
                         (lhs, rhs), Datatype(Vector(Float, r1))
                    | (Datatype(Matrix(Int,r1,c1)), Datatype(Matrix(Int,r2,c2))) ->
                        (lhs, rhs), Datatype(Matrix(Int, r1, c2))
                    | (Datatype(Matrix(Float,r1,c1)), Datatype(Matrix(Float,r2,c2))) ->
                        (lhs, rhs), Datatype(Matrix(Float, r1, c2))
                    | _                                  -> raise(Exceptions.IllegalCast)
                in

                let (e1ll, e2ll), d = cast e1' e2' type1 type2 in
                    let check_binop_type d =
                        match d with
                            Datatype(Int)               -> int_bops op e1ll e2ll
                            | Datatype(Float)           -> float_bops op e1ll e2ll
                            | Datatype(Bool)            -> bool_bops op e1ll e2ll
                            | Datatype(Vector(Int,n))   ->
                                let n_i = (match n with Int_lit(n1) -> n1 | _ -> -1) in
                                    vector_bops "int" n_i n op e1 e2
                            | Datatype(Vector(Float,n)) ->
                                let n_i = (match n with Int_lit(n1) -> n1 | _ -> -1) in
                                    vector_bops "float" n_i n op e1 e2
                            | Datatype(Matrix(Int,r,c)) ->
                                let r_i = (match r with Int_lit(n) -> n | _ -> -1)
                                and c_i = (match c with Int_lit(n) -> n | _ -> -1) in
                                    matrix_bops "int" r_i c_i r c op e1 e2
                            | Datatype(Matrix(Float,r,c)) ->
                                let r_i = (match r with Int_lit(n) -> n | _ -> -1)
                                and c_i = (match c with Int_lit(n) -> n | _ -> -1) in
                                    matrix_bops "float" r_i c_i r c op e1 e2
                            | _                         -> raise(Exceptions.UnsupportedBinopType)
                in
                check_binop_type d
            | S.SUnop(op, e, d)         ->
                let e' = expr builder e in
                let int_unops op =
                    (match op with
                        A.Neg     -> L.build_neg e' "tmp" builder
                        | A.Inc   -> L.build_store (L.build_add e' (L.const_int i32_t 1) "tmp"
builder) (lookup (match e with S.SId(s, d) -> s | _->raise(Exceptions.IncMustBeCalledOnID))) builder
                        | A.Dec   -> L.build_store (L.build_sub e' (L.const_int i32_t 1) "tmp"
builder) (lookup (match e with S.SId(s, d) -> s | _->raise(Exceptions.DecMustBeCalledOnID))) builder
```

```
                    | _       -> raise(Exceptions.IllegalIntUnop))
                in
                let float_unops op =
                    match op with
                        A.Neg   -> L.build_fneg e' "tmp" builder
                        | _     -> raise(Exceptions.IllegalFloatUnop)
                in
                let bool_unops op =
                    match op with
                        A.Not   -> L.build_not e' "tmp" builder
                        | _       -> raise(Exceptions.IllegalBoolUnop)
                in
                let check_unop_type d =
                    match d with
                        Datatype(Int)   -> int_unops op
                        | Datatype(Float) -> float_unops op
                        | Datatype(Bool)  -> bool_unops op
                        | _               -> raise(Exceptions.InvalidUnopType)
                in
                check_unop_type d
            | S.SRows(r)                -> L.const_int i32_t r
            | S.SCols(c)                -> L.const_int i32_t c
            | S.SLen(l)                 -> L.const_int i32_t l
            | S.STranspose(s,d)         ->
                (match d with
                    Datatype(Matrix(Int, c, r)) ->
                        let r_tr = (match c with Int_lit(n) -> n | _ -> -1) in
                        let c_tr = (match r with Int_lit(n) -> n | _ -> -1) in
                        let tmp_tr = L.build_alloca (array_t (array_t i32_t c_tr) r_tr) "tmpmat"
builder in
                        for i=0 to (r_tr-1) do
                            for j=0 to (c_tr-1) do
                                let mtr = build_matrix_access r_tr c_tr s (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                                let ld = L.build_gep tmp_tr [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                                ignore(build_store mtr ld builder);
                            done
                        done;
                        L.build_load (L.build_gep tmp_tr [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
                    | _ -> const_int i32_t 0)
            | S.SCall ("print_string", [e], d) ->
                L.build_call printf_func [| string_format_str ; (expr builder e) |] "printf" builder
            | S.SCall ("print_int", [e], d) ->
                L.build_call printf_func [| int_format_str ; (expr builder e) |] "printf" builder
            | S.SCall ("print_float", [e], d) ->
                L.build_call printf_func [| float_format_str ; (expr builder e) |] "printf" builder
            | S.SCall (f, act, d) ->
                let (fdef, fdecl) = StringMap.find f function_decls in
                let actuals = List.rev (List.map (expr builder) (List.rev act)) in
                let result =
                    (match fdecl.S.sreturn_type with
                        A.Datatype(A.Void) -> ""
```

```
                    | _ -> f ^ "_result") in
                L.build_call fdef (Array.of_list actuals) result builder
            | S.SNull                   -> L.const_null i32_t
            | S.SMatrix_access (s, se1, se2, d) ->
                let i = expr builder se1 and j = expr builder se2 in
                let access_i = (match se1 with S.SNum_lit(SInt_lit(n)) -> n | _ -> -1)
                and access_j = (match se2 with S.SNum_lit(SInt_lit(n)) -> n | _ -> -1) in
                    (build_matrix_access access_i access_j s (L.const_int i32_t 0) i j builder false)
            | S.SVector_access (s, se, d) ->
                let i = expr builder se in
                let access_i = (match se with S.SNum_lit(SInt_lit(n)) -> n | _ -> -1) in
                    (build_vector_access access_i s (L.const_int i32_t 0) i builder false)
            | S.SMatrix_lit (sll, d) ->
                (match d with
                    A.Datatype(A.Float) ->
                        let realOrder       = List.map List.rev sll in
                        let i64Lists        = List.map (List.map (expr builder)) realOrder in
                        let listOfArrays    = List.map Array.of_list i64Lists in
                        let i64ListOfArrays = List.rev (List.map (L.const_array float_t)
listOfArrays) in
                        let arrayOfArrays   = Array.of_list i64ListOfArrays in
                            L.const_array (array_t float_t (List.length (List.hd sll))) arrayOfArrays
                    | A.Datatype(A.Int)  ->
                        let realOrder       = List.map List.rev sll in
                        let i32Lists        = List.map (List.map (expr builder)) realOrder in
                        let listOfArrays    = List.map Array.of_list i32Lists in
                        let i32ListOfArrays = List.rev (List.map (L.const_array i32_t) listOfArrays)
in
                        let arrayOfArrays   = Array.of_list i32ListOfArrays in
                            L.const_array (array_t i32_t (List.length (List.hd sll))) arrayOfArrays
                    | _ -> raise(Exceptions.UnsupportedMatrixType))
            | S.SVector_lit (sl, d) ->
                (match d with
                    A.Datatype(A.Float) ->
                        let realOrder = List.rev sl in
                        let i64List = List.map (expr builder) realOrder in
                        let arrayOfi64 = Array.of_list i64List in
                            L.const_array (array_t float_t (List.length sl)) arrayOfi64
                    | A.Datatype(A.Int) ->
                        let realOrder = List.rev sl in
                        let i32List = List.map (expr builder) realOrder in
                        let arrayOfi32 = Array.of_list i32List in
                            L.const_array (array_t i32_t (List.length sl)) arrayOfi32
                    | _ -> raise(Exceptions.UnsupportedVectorType))
            | S.SMatrix_row (s, r, d) ->
                let access_i = (match r with SNum_lit(SInt_lit(n)) -> n | _-> -1) in
                let rows = L.array_length (L.type_of (L.build_load (L.build_gep (lookup s) [|
L.const_int i32_t 0 |] s builder) s builder)) in
                if (rows < access_i) then raise(Exceptions.MatrixOutOfBoundsAccess(""));
                L.build_load (L.build_gep (lookup s) [| L.const_int i32_t 0; L.const_int i32_t
access_i |] s builder) s builder
            | S.SMatrix_col (s, c, d) ->
                let c_i = (match c with SNum_lit(SInt_lit(n)) -> n | _ -> -1) in
                (match d with
```

```
                    Datatype(Matrix(Int,_,_)) ->
                        let rows = L.array_length (L.type_of (L.build_load (L.build_gep (lookup s) [|
L.const_int i32_t 0 |] s builder) s builder)) in
                        let cols = L.array_length (L.type_of (L.build_load (L.build_gep (lookup s) [|
L.const_int i32_t 0; L.const_int i32_t 0 |] s builder) s builder)) in
                        let tmp_m = L.build_alloca (array_t i32_t cols) "tmpmat" builder in
                        for j=0 to (rows-1) do
                            let m1 = build_matrix_access j c_i s (L.const_int i32_t 0) (L.const_int
i32_t j) (L.const_int i32_t c_i) builder false in
                            let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t j|]
"tmpmat" builder in
                            ignore(build_store m1 ld builder);
                        done;
                        L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
                    | Datatype(Matrix(Float,_,_)) ->
                        let rows = L.array_length (L.type_of (L.build_load (L.build_gep (lookup s) [|
L.const_int i32_t 0 |] s builder) s builder)) in
                        let cols = L.array_length (L.type_of (L.build_load (L.build_gep (lookup s) [|
L.const_int i32_t 0; L.const_int i32_t 0 |] s builder) s builder)) in
                        let tmp_m = L.build_alloca (array_t float_t cols) "tmpmat" builder in
                        for j=0 to (rows-1) do
                            let m1 = build_matrix_access j c_i s (L.const_int i32_t 0) (L.const_int
i32_t j) (L.const_int i32_t c_i) builder false in
                            let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t j|]
"tmpmat" builder in
                            ignore(build_store m1 ld builder);
                        done;
                        L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
                    | _ -> L.const_int i32_t 0)
        in

        let add_terminal builder f =
            match L.block_terminator (L.insertion_block builder) with
                Some _ -> ()
                | None -> ignore (f builder)
        in

        let rec stmt builder = function
            S.SBlock sl -> List.fold_left stmt builder sl
            | S.SExpr e -> ignore (expr builder e); builder
            | S.SReturn e ->
                ignore(match fdecl.S.sreturn_type with
                    A.Datatype(A.Void)  -> L.build_ret_void builder
                    | _                 -> L.build_ret (expr builder e) builder); builder
            | S.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 then_bb = L.append_block context
                    "then" the_function in
                add_terminal
                    (stmt (L.builder_at_end context then_bb) then_stmt)
```

```
                (L.build_br merge_bb);
            let else_bb = L.append_block context
                "else" the_function in
            add_terminal
                (stmt (L.builder_at_end context else_bb) else_stmt)
                (L.build_br merge_bb);
            ignore (L.build_cond_br bool_val then_bb else_bb builder);
            L.builder_at_end context merge_bb
        | S.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
        | S.SFor (e1, e2, e3, body) -> stmt builder
            (S.SBlock [S.SExpr e1 ;
                S.SWhile (e2, S.SBlock [body ;
                    S.SExpr e3]) ])
    in

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

    (* Add a return if the last block falls off the end *)
    add_terminal builder
        (match fdecl.S.sreturn_type with
            A.Datatype(A.Void) -> L.build_ret_void;
            | t -> L.build_ret (L.const_int (ltype_of_datatype t) 0))
in
List.iter build_function_body functions;

the_module (*returned as a module to whatever called this*)
```

A.3.4 plt/src/exceptions.ml

```
(*
 * COMS4115: CMAT Exceptions
 *
 * Authors:
 *  - Marissa Ojeda
 *  - Daniel Rojas
 *  - Mike Berkowitz
 *  - Frank Cabada
 *)

(* Compiler Exceptions *)
```

```
exception NoFileArgument

(* Analyzer Exceptions *)
exception AllVoidFunctionsMustNotReturn of string
exception AllNonVoidFunctionsMustEndWithReturn of string
exception AssignmentTypeMismatch of string * string
exception CannotUseRowsOnNonMatrix of string
exception CannotUseTransposeOnNonMatrix of string
exception CannotUseColsOnNonMatrix of string
exception CannotUseLenOnNonVector of string
exception DuplicateGlobal of string
exception DuplicateFuncOrLocal of string
exception FunctionNotFound of string
exception IncorrectNumberOfArguments of string * int * int
exception InvalidBinopExpression of string
exception InvalidUnaryOperation
exception MalformedMatrixLit
exception MatrixDimensionMustBeInt
exception MatrixAccessOnNonMatrix of string
exception MatrixColOnNonMatrix of string
exception MatrixLitMustBeOneType
exception VectorLitMustBeOneType
exception MatrixOutOfBoundsAccess of string
exception MatrixRowOnNonMatrix of string
exception MismatchedMatricesForAddSub of string
exception MismatchedMatricesForMult of string
exception MismatchedVectorsForBinop of string
exception ReturnTypeMismatch of string * string
exception UndefinedID of string
exception UnsupportedMatrixBinop of string
exception UnsupportedStringBinop of string
exception UnsupportedVectorBinop of string
exception VectorAccessOnNonMatrix of string
exception VectorDimensionMustBeIntLit
exception VoidFunctionFormal of string
exception VoidFunctionLocal of string
exception VoidFunc of string
exception VoidGlobal of string

(* Codegen Exceptions *)
exception AssignLHSMustBeAssignable
exception DecMustBeCalledOnID
exception IllegalBoolBinop
exception IllegalBoolUnop
exception IllegalIntUnop
exception IllegalCast
exception IllegalFloatBinop
exception IllegalFloatUnop
exception IllegalIntBinop
exception IllegalMatrixBinop
exception IllegalVectorBinop
exception IncMustBeCalledOnID
exception InvalidMatrixDimension
exception InvalidUnopType
```

```
exception InvalidVectorDimension
exception MatrixOutOfBoundsAccess of string
exception UnsupportedBinopType
exception UnsupportedMatrixType
exception UnsupportedVectorType
exception VectorOutOfBoundsAccess of string
```

A.3.5 plt/src/parser.mly

```
/*
 * COMS4115: CMAT Parser
 *
 * Authors:
 *  - Marissa Ojeda
 *  - Daniel Rojas
 *  - Mike Berkowitz
 *  - Frank Cabada
 */

%{ open Ast %}

/* Delimiters */
%token LPAREN RPAREN LBRACE RBRACE LBRACKET RBRACKET

/* Control Flow */
%token IF ELSE WHILE FOR RETURN

/* Conditionals */
%token EQ NEQ LT GT LEQ GEQ AND OR NOT

/* Arithmetic */
%token PLUS MINUS TIMES DIVIDE ASSIGN INC DEC

/* Types */
%token INT BOOL VOID STRING FLOAT TRUE FALSE MATRIX VECTOR

/* Misc */
%token SEMI COMMA COLON ROWS COLS LEN TRANSPOSE BAR

/* Literals, identifiers, EOF */
%token <Ast.num> NUM_LIT
%token <string> ID
%token <string> STRING_LIT
%token NULL
%token EOF

/* Precedence and associativity of each operator */
%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left OR
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
```

```
%left PLUS MINUS INC DEC
%left TIMES DIVIDE
%right NOT NEG

%start program
%type <Ast.program> program

%%

program:
    decls EOF { $1 }

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

fdecl:
  datatype ID LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list RBRACE
    { { return_type = $1; fname = $2; formals = $4;
      locals = List.rev $7; body = List.rev $8 } }

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

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

datatype:
  primitives { Datatype($1) }

primitives:
    INT                                                    { Int }
  | BOOL                                                   { Bool }
  | VOID                                                   { Void }
  | FLOAT                                                  { Float }
  | STRING                                                 { String }
  | VECTOR primitives LBRACKET NUM_LIT RBRACKET            { Vector($2, $4) }
  | MATRIX primitives LBRACKET NUM_LIT COMMA NUM_LIT RBRACKET   { Matrix($2, $4, $6) }

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

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

gdecl:
    datatype ID SEMI { ($1, $2) }

stmt_list:
    /* nothing */  { [] }
```

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

stmt:
    expr SEMI                                                    { Expr $1 }
  | RETURN SEMI                                                  { Return Noexpr }
  | RETURN expr 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:
    NUM_LIT                                                      { Num_lit($1) }
  | STRING_LIT                                                   { String_lit($1) }
  | TRUE                                                         { Bool_lit(true) }
  | FALSE                                                        { Bool_lit(false) }
  | NULL                                                         { Null }
  | ID                                                           { Id($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 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 NEG                                         { Unop(Neg, $2) }
  | NOT expr                                                     { Unop(Not, $2) }
  | INC expr                                                     { Unop(Inc, $2) }
  | DEC expr                                                     { Unop(Dec, $2) }
  | expr ASSIGN expr                                             { Assign($1, $3) }
  | LPAREN expr RPAREN                                           { $2 }
  | ID LPAREN actuals_opt RPAREN                                 { Call($1, $3) }
  | BAR vect_lit BAR                                             { Vector_lit($2) }
  | LBRACKET mat_lit RBRACKET                                    { Matrix_lit($2) }
  | ID LBRACKET expr RBRACKET                                    { Vector_access($1, $3) }
  | ID LBRACKET expr COMMA expr RBRACKET                         { Matrix_access($1, $3, $5) }
  | ID LBRACKET expr COMMA COLON RBRACKET                        { Matrix_row($1, $3) }
  | ID LBRACKET COLON COMMA expr RBRACKET                        { Matrix_col($1, $5) }
  | ID COLON ROWS                                                { Rows($1) }
  | ID COLON COLS                                                { Cols($1) }
  | ID COLON LEN                                                 { Len($1) }
  | ID COLON TRANSPOSE                                           { Transpose($1) }

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

actuals_opt:
```

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

lit:
    NUM_LIT                       { $1 }

vect_lit:
      lit                         { [$1] }
    | vect_lit BAR lit            { $3 :: $1 }

mat_lit:
    lit_list                      { [$1] }
    | mat_lit SEMI lit_list       { $3 :: $1 }

lit_list:
    lit                           { [$1] }
    | lit_list COMMA lit          { $3 :: $1 }

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

A.3.6 plt/src/prep.ml

```
(*
 * COMS4115: CMAT Prep Module
 *
 * Authors:
 *  - Marissa Ojeda
 *  - Daniel Rojas
 *  - Mike Berkowitz
 *  - Frank Cabada
 *)

let process filename =
    let rec read_file filename =
      let file_regex = Str.regexp "<.+\\.cmat"  in
      let inc_regex = Str.regexp "#include[ ]+<.+>;" in
      let has_file l =
        let has_inc l =
          try ignore (Str.search_forward inc_regex l 0); true
          with Not_found -> false
        in
        if has_inc l then
          try ignore(Str.search_forward file_regex l 0); true
          with Not_found -> false
        else false
      in
    let lines = ref [] in
    let ic = open_in filename in
    try
      while true; do
        let l = input_line ic in
        let l = (if (has_file l) then ( Str.replace_first inc_regex
```

```
        (read_file (Str.string_after (Str.matched_string l) 1) )l ) else l) in
        lines:=  l :: !lines;
      done;
      String.concat "" (List.map (fun i -> i ^ String.make 1 '\n') (List.rev !lines))
    with End_of_file -> ignore(close_in ic);
    String.concat "" (List.map (fun i -> i ^ String.make 1 '\n') (List.rev !lines))
in read_file filename
```

A.3.7 plt/src/sast.ml

```
(*
 * COMS4115: CMAT SAST
 *
 * Authors:
 *  - Marissa Ojeda
 *  - Daniel Rojas
 *  - Mike Berkowitz
 *  - Frank Cabada
 *)

open Ast

type snum =
              SInt_lit of int
       |      SFloat_lit of float

(* Expressions *)
type sexpr =
         SNum_lit of snum
       | SBool_lit of bool
       | SString_lit of string
       | SMatrix_lit of sexpr list list * datatype
       | SVector_lit of sexpr list * datatype
       | SId of string * datatype
       | SNoexpr
       | SNull
       | SBinop of sexpr * op * sexpr * datatype
       | SUnop of uop * sexpr * datatype
       | SAssign of sexpr * sexpr * datatype
       | SCall of string * sexpr list * datatype
       | SVector_access of string * sexpr * datatype
       | SMatrix_access of string * sexpr * sexpr * datatype
       | SMatrix_row of string * sexpr * datatype
       | SMatrix_col of string * sexpr * datatype
       | SRows of int
       | SCols of int
       | SLen of int
       | STranspose of string * datatype

(* Statements *)
type sstmt =
         SBlock of sstmt list
       | SExpr of sexpr
       | SIf of sexpr * sstmt * sstmt
```

```
       | SFor of sexpr * sexpr * sexpr * sstmt
       | SWhile of sexpr * sstmt
       | SReturn of sexpr

(* Function Declarations *)
type sfunc_decl = {
       sreturn_type   : datatype;
       sfname                 : string;
       sformals               : formal list;
       slocals                : local list;
       sbody                  : sstmt list;
}

(* All method declarations | Main entry method *)
type sprogram = var_dec list * func_decl list
```

A.3.8 plt/src/scanner.mll

```
(*
 * COMS4115: CMAT Scanner
 *
 * Authors:
 *  - Marissa Ojeda
 *  - Daniel Rojas
 *  - Mike Berkowitz
 *  - Frank Cabada
 *)

{ open Parser }

rule token = parse
(* Whitespace *)
       [' ' '\t' '\r' '\n'] { token lexbuf }

(* Comments *)
| "/*" { comment lexbuf }

(* Delimiters *)
| '(' { LPAREN }  | ')' { RPAREN } | '{' { LBRACE } | '}' { RBRACE }
| '[' { LBRACKET } | ']' { RBRACKET }

(* Control Flow *)
| "if" { IF } | "else" { ELSE } | "while" { WHILE }
| "for" { FOR } | "return" { RETURN }

(* Conditionals *)
| "==" { EQ } | "!=" { NEQ } | '<' { LT } | ">" { GT }
| "<=" { LEQ } | ">=" { GEQ } | "&&" { AND } | "||" { OR } | '!' { NOT }

(* Arithmetic *)
| '+' { PLUS } | '-' { MINUS } | '*' { TIMES } | '/' { DIVIDE }
| '=' { ASSIGN } | "++" { INC } | "--" { DEC }

(* Types *)
```

```
| "int" { INT } | "float" { FLOAT } | "bool" { BOOL } | "void" { VOID }
| "String" { STRING } | "true" { TRUE } | "false" { FALSE }
| "matrix" { MATRIX } | "vector" { VECTOR }

(* Misc. *)
| ';' { SEMI } | ',' { COMMA } | ':' { COLON } | '|' { BAR }
| "rows" { ROWS } | "cols" { COLS } | "len" { LEN } | "tr" { TRANSPOSE }

(* Literals *)
| ['0'-'9']+ as lxm { NUM_LIT(Ast.Int_lit(int_of_string lxm)) }
| ['0'-'9']* '.' ['0'-'9']+ as lxmd { NUM_LIT(Ast.Float_lit(float_of_string lxmd)) }
| '"' (([^ '"'] | "\\\"")* as strlit) '"' { STRING_LIT(strlit) }
| "null" { NULL }

(* Identifiers, EOF *)
| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']* as lxm { ID(lxm) }
| eof { EOF }
| _ as char { raise (Failure("illegal character " ^
                                                    Char.escaped char)) }

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

A.3.9 plt/src/semant.ml

```
(*
 * COMS4115: CMAT Semantic Analyzer
 *
 * Authors:
 *  - Marissa Ojeda
 *  - Daniel Rojas
 *  - Mike Berkowitz
 *  - Frank Cabada
 *)

open Ast
open Sast
open Exceptions
open Utils

module StringMap = Map.Make(String)

let get_equality_binop_type type1 type2 se1 se2 op =
       if (type1 = Datatype(Float) || type2 = Datatype(Float)) then
raise(Exceptions.InvalidBinopExpression "Cannot use equality for floats")
       else
              if type1 = type2 && (type1 = Datatype(String) || type1 = Datatype(Int)) then
SBinop(se1, op, se2, type1)
              else raise (Exceptions.InvalidBinopExpression("Can only use equality operators with
ints and Strings"))

let get_logical_binop_type se1 se2 op = function
              | (Datatype(Bool), Datatype(Bool)) -> SBinop(se1, op, se2, Datatype(Bool))
```

```
              | (Datatype(Int), Datatype(Int))   -> SBinop(se1, op, se2, Datatype(Int))
              | (Datatype(Int), Datatype(Bool))  -> SBinop(se1, op, se2, Datatype(Bool))
              | (Datatype(Bool), Datatype(Int))  -> SBinop(se1, op, se2, Datatype(Bool))
              | _ -> raise (Exceptions.InvalidBinopExpression "Can only use Bools/Ints for logical
operators")

let get_arithmetic_binop_type se1 se2 op = function
                (Datatype(Int), Datatype(Float))
              | (Datatype(Float), Datatype(Int))
              | (Datatype(Float), Datatype(Float))       -> SBinop(se1, op, se2, Datatype(Float))
              | (Datatype(String), Datatype(String))     ->
                     (match op with
                            Add -> SBinop(se1, op, se2, Datatype(String))
                            | _ -> raise(Exceptions.UnsupportedStringBinop("Cannot subtract,
multiply, or divide strings")))
              | (Datatype(Int), Datatype(Int))           -> SBinop(se1, op, se2, Datatype(Int))
              | (Datatype(Vector(typ1, n1)), Datatype(Vector(typ2, n2))) ->
                     (match op with
                            Add | Sub      ->
                                   if typ1=typ2 && n1=n2 then
                                          SBinop(se1, op, se2, Datatype(Vector(typ1, n1)))
                                   else raise(Exceptions.MismatchedVectorsForBinop("Vectors must be
same type and size for +/-"))
                                | _            -> raise(Exceptions.UnsupportedVectorBinop("Cannot
multiply or divide vectors")))
              | (Datatype(Matrix(typ1, i1, j1)), Datatype(Vector(typ2, n))) ->
                     (match op with
                            | Mult         ->
                                   if typ1=typ2 && j1 = n then
                                          SBinop(se1, op, se2, Datatype(Vector(typ1, i1)))
                                   else raise(Exceptions.MismatchedMatricesForMult("Matrix M1(i,j)
and vector V(n) must have j = n and be of same type to be multiplied"))
                                | _ -> raise(Exceptions.UnsupportedMatrixBinop("Cannot add. subtract,
divide matrices with vectors")))
              | (Datatype(Matrix(typ1, i1, j1)), Datatype(Matrix(typ2, i2, j2))) ->
                     (match op with
                            Add | Sub      ->
                                   if typ1=typ2 && i1=i2 && j1=j2 then
                                          SBinop(se1, op, se2, Datatype(Matrix(typ1, i1, j2)))
                                   else raise(Exceptions.MismatchedMatricesForAddSub("Matrices must
be same type and dimensions for +/-"))
                                | Mult         ->
                                   if typ1=typ2 && j1 = i2 then
                                          SBinop(se1, op, se2, Datatype(Matrix(typ1, i1, j2)))
                                   else raise(Exceptions.MismatchedMatricesForMult("Matrices
M1(i1,j1) and M2(i2,j2) must have j1 = i2 and be of same type to be multiplied"))
                                | _ -> raise(Exceptions.UnsupportedMatrixBinop("Cannot divide
matrices")))
              | (Datatype(Int), Datatype(Vector(Int, n))) ->
                     (match op with
                            Mult -> SBinop(se1, op, se2, Datatype(Vector(Int, n)))
                            | _ -> raise(Exceptions.UnsupportedVectorBinop("Cannot add, subtract, or
divide ints with vectors")))
              | (Datatype(Int), Datatype(Matrix(Int,i,j))) ->
```

```
                        (match op with
                                Mult -> SBinop(se1, op, se2, Datatype(Matrix(Int, i, j)))
                                | _ -> raise(Exceptions.UnsupportedMatrixBinop("Cannot add, subtract, or
divide ints with matrices")))
                | (Datatype(Float), Datatype(Vector(Float, n))) ->
                        (match op with
                                Mult -> SBinop(se1, op, se2, Datatype(Vector(Float, n)))
                                | _ -> raise(Exceptions.UnsupportedVectorBinop("Cannot add, subtract, or
divide floats with vectors")))
                | (Datatype(Float), Datatype(Matrix(Float,i,j))) ->
                        (match op with
                                Mult -> SBinop(se1, op, se2, Datatype(Matrix(Float, i, j)))
                                | _ -> raise(Exceptions.UnsupportedMatrixBinop("Cannot add, subtract, or
divide floats with matrices")))
                | _ -> raise (Exceptions.InvalidBinopExpression("Arithmetic operators on unsupported
type"))

let rec get_ID_type s func_st =
        try StringMap.find s func_st
        with | Not_found -> raise (Exceptions.UndefinedID(s))

and check_assign fname_map func_st e1 e2 =
        let se1 = expr_to_sexpr fname_map func_st e1 in
        let type1 = get_type_from_sexpr se1 in
        let se2 = expr_to_sexpr fname_map func_st e2 in
        let type2 = get_type_from_sexpr se2 in
        match type1, type2 with
                Datatype(String), Datatype(Int)
                | Datatype(Int), Datatype(String) -> SAssign(se1, se2, type1)
                | Datatype(Int), Datatype(Bool) -> SAssign(se1, se2, type1)
                | Datatype(Bool), Datatype(Int) -> SAssign(se1, se2, type1)
                | Datatype(Vector(dv,n)), Datatype(Matrix(dm,r,Int_lit(1))) -> if ((dv=dm) && (n=r))
then SAssign(se1, se2, type1) else raise(Exceptions.AssignmentTypeMismatch(Utils.string_of_datatype
type1, Utils.string_of_datatype type2))
                | Datatype(Vector(dv,n)), Datatype(Matrix(dm,Int_lit(1),c)) -> if ((dv=dm) && (n=c))
then SAssign(se1, se2, type1) else raise(Exceptions.AssignmentTypeMismatch(Utils.string_of_datatype
type1, Utils.string_of_datatype type2))
                | _ ->
        if type1 = type2
                then SAssign(se1, se2, type1)
        else raise(Exceptions.AssignmentTypeMismatch(Utils.string_of_datatype type1,
Utils.string_of_datatype type2))

and check_unop fname_map func_st op e =
        let check_num_unop t = function
                  Neg -> t
                | Inc -> t
                | Dec -> t
                |       _               -> raise(Exceptions.InvalidUnaryOperation)
        in
        let check_bool_unop x = match x with
                        Not     -> Datatype(Bool)
                |       _               -> raise(Exceptions.InvalidUnaryOperation)
        in
```

```
        let se = expr_to_sexpr fname_map func_st e in
        let t = get_type_from_sexpr se in
                match t with
                  Datatype(Int)
                | Datatype(Float) -> SUnop(op, se, check_num_unop t op)
                | Datatype(Bool)  -> SUnop(op, se, check_bool_unop op)
                | _                     -> raise(Exceptions.InvalidUnaryOperation)

and check_binop fname_map func_st e1 op e2 =
        let se1 = expr_to_sexpr fname_map func_st e1 in
        let se2 = expr_to_sexpr fname_map func_st e2 in
        let type1 = get_type_from_sexpr se1 in
        let type2 = get_type_from_sexpr se2 in
        match op with
        Equal | Neq -> get_equality_binop_type type1 type2 se1 se2 op
        | And | Or -> get_logical_binop_type se1 se2 op (type1, type2)
        | Less | Leq | Greater | Geq when type1 = type2 && (type1 = Datatype(Int) || type1 =
Datatype(Float)) -> SBinop(se1, op, se2, type1)
        | Add | Mult | Sub | Div -> get_arithmetic_binop_type se1 se2 op (type1, type2)
        | _ -> raise (Exceptions.InvalidBinopExpression ((Utils.string_of_op op) ^ " is not a
supported binary op"))

and check_expr_is_int func_st e = match e with
        Num_lit(Int_lit(n)) -> Datatype(Int)
        | Id(s)                      -> get_ID_type s func_st
        | _ -> raise(Exceptions.MatrixDimensionMustBeInt)

and check_matrix_row fname_map func_st s e =
        ignore(check_expr_is_int func_st e);
        let t = get_ID_type s func_stin
                match t with
                        Datatype(Matrix(d,r,c)) -> SMatrix_row(s, expr_to_sexpr fname_map func_st e,
Datatype(Matrix(d,r,Int_lit(1))))
                        | _ -> raise(Exceptions.MatrixRowOnNonMatrix(s))

and check_matrix_col fname_map func_st s e =
        ignore(check_expr_is_int func_st e);
        let t = get_ID_type s func_stin
                match t with
                        Datatype(Matrix(d,r,c)) -> SMatrix_col(s, expr_to_sexpr fname_map func_st e,
Datatype(Matrix(d,Int_lit(1),c)))
                        | _ -> raise(Exceptions.MatrixColOnNonMatrix(s))

and check_matrix_access fname_map func_st s e1 e2 =
        ignore(check_expr_is_int func_st e1);
        ignore(check_expr_is_int func_st e2);
        let t = get_ID_type s func_stin
                match t with
                        Datatype(Matrix(d,rows,cols)) ->
                                SMatrix_access(s, expr_to_sexpr fname_map func_st e1, expr_to_sexpr
fname_map func_st e2, Datatype(d))
                        | _ -> raise(Exceptions.MatrixAccessOnNonMatrix(s))

and check_vector_access fname_map func_st s e =
```

```
	ignore(check_expr_is_int func_st e);
	let t = get_ID_type s func_stin
		match t with
			Datatype(Vector(d,_)) -> SVector_access(s, expr_to_sexpr fname_map func_st e,
Datatype(d))
			| _ -> raise(Exceptions.VectorAccessOnNonMatrix(s))

and lit_to_slit n = match n with
	Int_lit(n) -> SNum_lit(SInt_lit(n))
	| Float_lit(n) -> SNum_lit(SFloat_lit(n))

and typ_of_lit n = match n with
	Int_lit(n) -> Datatype(Int)
	| Float_lit(n) -> Datatype(Float)

and check_matrix_lit fname_map func_st nll =
	let snll = (List.map (fun nl -> (List.map lit_to_slit nl)) nll) in
	let first = List.hd (List.hd nll) in
	let first_size = List.length (List.hd nll) in
		ignore(List.iter (fun nl -> if (List.length nl = first_size) then () else
raise(Exceptions.MalformedMatrixLit)) nll);
	let first_typ = typ_of_lit first in
		ignore(List.iter (fun nl -> List.iter (fun n ->
			(let typ = typ_of_lit n in
				if (typ = first_typ)
					then ()
					else raise(Exceptions.MatrixLitMustBeOneType))) nl) nll);
	SMatrix_lit(snll, first_typ)

and check_vector_lit fname_map func_st nl =
	let snl = (List.map lit_to_slit nl) in
	let first = (List.hd nl) in
	let first_typ = typ_of_lit first in
		ignore(List.iter (fun n ->
			(let typ = typ_of_lit n in
				if (typ = first_typ)
					then ()
				else raise(Exceptions.VectorLitMustBeOneType))) nl);
	SVector_lit(snl, first_typ)

and function_decl s fname_map =
	try StringMap.find s fname_map
	with Not_found -> raise (Exceptions.FunctionNotFound(s))

and check_rows s func_st =
	let typ = get_ID_type s func_st in
		match typ with
			Datatype(Matrix(_, r, _)) -> (match r with Int_lit(n) -> SRows(n) | _ ->
raise(Exceptions.MatrixDimensionMustBeInt))
			| _ -> raise(Exceptions.CannotUseRowsOnNonMatrix(s))

and check_cols s func_st =
	let typ = get_ID_type s func_st in
		match typ with
```

```
                    Datatype(Matrix(_, _, c)) -> (match c with Int_lit(n) -> SCols(n) | _ ->
raise(Exceptions.MatrixDimensionMustBeInt))
                    | _ -> raise(Exceptions.CannotUseColsOnNonMatrix(s))

and check_transpose s func_st =
       let typ = get_ID_type s func_st in
              match typ with
                    Datatype(Matrix(d, r, c)) -> STranspose(s, Datatype(Matrix(d,c,r)))
                    | _ -> raise(Exceptions.CannotUseTransposeOnNonMatrix(s))

and check_len s func_st =
       let typ = get_ID_type s func_st in
              (match typ with
                    Datatype(Vector(_, l)) -> (match l with Int_lit(n) -> SLen(n) | _ ->
raise(Exceptions.VectorDimensionMustBeIntLit))
                    | _ -> raise(Exceptions.CannotUseLenOnNonVector(s)))

and expr_to_sexpr fname_map func_st = function
         Num_lit(Int_lit(n))              -> SNum_lit(SInt_lit(n))
       | Num_lit(Float_lit(n))            -> SNum_lit(SFloat_lit(n))
       | Bool_lit(b)                -> SBool_lit(b)
       | String_lit(s)                    -> SString_lit(s)
       | Id(s)                            -> SId(s, get_ID_type s func_st)
       | Null                             -> SNull
       | Noexpr                           -> SNoexpr
       | Unop(op, e)                      -> check_unop fname_map func_st op e
       | Assign(s, e)                     -> check_assign fname_map func_st s e
       | Binop(e1, op, e2)                -> check_binop fname_map func_st e1 op e2
       | Call(s, el)                      -> let fd = function_decl s fname_map in
              if List.length el != List.length fd.formals then
                    raise (Exceptions.IncorrectNumberOfArguments(fd.fname, List.length el,
List.length fd.formals))
              else
              SCall(s, List.map (expr_to_sexpr fname_map func_st) el, fd.return_type)
       | Vector_access(s, e)        -> check_vector_access fname_map func_st s e
       | Matrix_access(s, e1, e2)   -> check_matrix_access fname_map func_st s e1 e2
       | Matrix_row(s, e)           -> check_matrix_row fname_map func_st s e
       | Matrix_col(s, e)           -> check_matrix_col fname_map func_st s e
       | Matrix_lit(nll)                  -> check_matrix_lit fname_map func_st nll
       | Vector_lit(nl)           -> check_vector_lit fname_map func_st nl
       | Rows(s)                                -> check_rows s func_st
       | Cols(s)                                -> check_cols s func_st
       | Len(s)                                 -> check_len s func_st
       | Transpose(s)                     -> check_transpose s func_st

and get_type_from_sexpr sexpr = match sexpr with
         SNum_lit(SInt_lit(_))                          -> Datatype(Int)
       | SNum_lit(SFloat_lit(_))                  -> Datatype(Float)
       | SBool_lit(_)                                 -> Datatype(Bool)
       | SString_lit(_)                               -> Datatype(String)
       | SNoexpr                                             -> Datatype(Void)
       | SNull                                               -> Datatype(Void)
       | SRows(r)                                            -> Datatype(Int)
       | SCols(c)                                            -> Datatype(Int)
```

```
        | SLen(l)                                                -> Datatype(Int)
        | STranspose(_,d)                                 -> d
        | SId(_, d)                                       -> d
        | SBinop(_, _, _, d)                        -> d
        | SAssign(_, _, d)                                -> d
        | SCall(_, _, d)                                  -> d
        | SUnop(_, _, d)                                  -> d
        | SVector_access(_, _, d)                   -> d
        | SMatrix_access(_, _, _, d)          -> d
        | SMatrix_row(_, _, d)                            -> d
        | SMatrix_col(_, _, d)                            -> d
        | SMatrix_lit(sll, d)                       ->
              let c = List.length (List.hd sll) in
              let r = List.length sll in
              (match d with
                     Datatype(Int)          -> Datatype(Matrix(Int, Int_lit(r), Int_lit(c)))
                     | Datatype(Float)      -> Datatype(Matrix(Float, Int_lit(r), Int_lit(c)))
                     | _                           -> raise(Exceptions.UnsupportedMatrixType))
        | SVector_lit (sl, d)                 ->
              let r = List.length sl in
              match d with
                       Datatype(Int)      -> Datatype(Vector(Int, Int_lit(r)))
                     | Datatype(Float)    -> Datatype(Vector(Float, Int_lit(r)))
                     | _                           -> raise(Exceptions.UnsupportedVectorType)

let add_reserved_functions =
       let reserved_stub name return_type formals =
              {
                     return_type    = return_type;
                     fname          = name;
                     formals        = formals;
                     locals         = [];
                     body           = [];
              }
       in
       let void_t = Datatype(Void) in
       let str_t = Datatype(String) in
       let i32_t = Datatype(Int) in
       let float_t = Datatype(Float) in
       let mf t n = Formal(t, n) in (* Make formal *)
       let reserved = [
              reserved_stub "print_string" (void_t)       ([mf str_t "string_in"]);
              reserved_stub "print_int"           (void_t)       ([mf i32_t "int_in"]);
              reserved_stub "print_float"         (void_t)       ([mf float_t "float_in"]);
       ] in
       reserved

(* Variable Declaration Checking Functions *)
let report_duplicate s li =
       let rec helper = function
              n1 :: n2 :: _ when n1 = n2 ->
                     if s = "global" then raise(Exceptions.DuplicateGlobal(n1))
                     else if s = "function" then raise(Exceptions.DuplicateFuncOrLocal(n1))
              | _ :: t -> helper t
```

```
		| [] -> ()
	in helper (List.sort compare li)

let check_not_void s var_decl =
	match var_decl with
		(Datatype(Void), n)   ->
			if s = "global" then raise(Exceptions.VoidGlobal(n))
			else if s = "function" then raise(Exceptions.VoidFunc(n))
		| _                                    -> ()

let check_not_void_formal form =
	match form with
		Formal(Datatype(Void), n)    -> raise(Exceptions.VoidFunctionFormal(n))
		| _                                          -> ()

let check_not_void_local local =
	match local with
		Local(Datatype(Void), n)     -> raise(Exceptions.VoidFunctionLocal(n))
		| _                                          -> ()

let add_to_global_symbol_table globs =
	List.fold_left
		(fun m (t,n) -> StringMap.add n t m) StringMap.empty globs

let get_global_id = function
	| (Datatype(p), n) -> n

let get_formal_id = function
	| Formal(Datatype(p), n) -> n

let get_formal_type = function
	| Formal(Datatype(p), n) -> Datatype(p)

let get_local_id = function
	| Local(Datatype(p), n) -> n

let get_local_type = function
	| Local(Datatype(p), n) -> Datatype(p)

let check_var_decls globals =
	ignore(List.iter (check_not_void "global") globals);
	ignore(report_duplicate "global" (List.map snd globals));
	add_to_global_symbol_table globals;;

(* Function Declaration Checking Functions *)
let fdecl_to_func_st globals fdecl =
	let ffunc_st = List.fold_left (fun m f -> StringMap.add (get_formal_id f) (get_formal_type f)
m) StringMap.empty fdecl.formals in
		let lffunc_st = List.fold_left (fun m l -> StringMap.add (get_local_id l)
(get_local_type l) m) ffunc_st fdecl.locals in
			List.fold_left (fun m g -> StringMap.add (snd g) (fst g) m) lffunc_st globals

let rec stmt_to_sstmt fname_map func_st = function
	 Return(e)                          -> SReturn(expr_to_sexpr fname_map func_st e)
```

```
	| Block(sl)                  -> SBlock(convert_stmt_list_to_sstmt_list fname_map func_st sl)
	| Expr(e)                          -> SExpr(expr_to_sexpr fname_map func_st e)
	| If(e, s1, s2)             -> SIf((expr_to_sexpr fname_map func_st e), (stmt_to_sstmt
fname_map func_st s1), (stmt_to_sstmt fname_map func_st s2))
	| For(e1, e2, e3, s)  -> SFor((expr_to_sexpr fname_map func_st e1), (expr_to_sexpr fname_map
func_st e2), (expr_to_sexpr fname_map func_st e3), (stmt_to_sstmt fname_map func_st s))
	| While(e, s)               -> SWhile((expr_to_sexpr fname_map func_st e), (stmt_to_sstmt
fname_map func_st s))

and convert_stmt_list_to_sstmt_list fname_map func_st stmt_list = List.map (stmt_to_sstmt fname_map
func_st) stmt_list

let fdecls_to_fname_map fdecls =
	List.fold_left
		(fun m fd -> StringMap.add fd.fname fd m) StringMap.empty (fdecls @
add_reserved_functions)

let convert_fdecl_to_sfdecl globs fname_map fdecl =
	{
		sfname                = fdecl.fname;
		sreturn_type   = fdecl.return_type;
		sformals              = fdecl.formals;
		slocals               = fdecl.locals;
		sbody                 = (convert_stmt_list_to_sstmt_list fname_map (fdecl_to_func_st
globs fdecl) fdecl.body);
	}

let check_function_return fname fbody returnType =
	let len = List.length fbody in
		if len > 0
			then let final_stmt = List.hd (List.rev fbody) in
				match returnType, final_stmt with
					Datatype(Void), Return(_) ->
raise(Exceptions.AllVoidFunctionsMustNotReturn(fname))
					| Datatype(Void), _     -> ()
					| _, Return(_)              -> ()
					| _, _                                  ->
raise(Exceptions.AllNonVoidFunctionsMustEndWithReturn(fname))
			else
				if returnType = Datatype(Void) then ()
				else raise(Exceptions.AllNonVoidFunctionsMustEndWithReturn(fname))

let check_return fname_map fdecl func_st e =
	let se = expr_to_sexpr fname_map func_st e in
	let t = get_type_from_sexpr se in
		(match fdecl.return_type with
			Datatype(Matrix(d,Int_lit(0),Int_lit(0))) ->
				(match t with
					Datatype(Matrix(d,_,_)) -> ()
					| _ ->
raise(Exceptions.ReturnTypeMismatch(Utils.string_of_datatype t, Utils.string_of_datatype
fdecl.return_type)))
			| _ -> if (t=fdecl.return_type) then () else
raise(Exceptions.ReturnTypeMismatch(Utils.string_of_datatype t, Utils.string_of_datatype
```

```
fdecl.return_type)))

let rec check_stmt globs fname_map fdecl = function
       Return(e)                                -> check_return fname_map fdecl (fdecl_to_func_st
globs fdecl) e
       | Block(sl)                        -> check_fbody globs fname_map fdecl sl
       | If(e, s1, s2)                    -> check_if globs fname_map fdecl s1 s2
       | While(e, s)                      -> check_while globs fname_map fdecl s
       | For(e1, e2, e3, s)         -> check_for globs fname_map fdecl s
       | Expr(e)                                -> ()

and check_fbody globs fname_map fdecl fbody =
       ignore(List.iter (check_stmt globs fname_map fdecl) fbody);

and check_if globs fname_map fdecl s1 s2 =
       ignore(check_stmt globs fname_map fdecl s1);
       ignore(check_stmt globs fname_map fdecl s2);

and check_while globs fname_map fdecl stmt =
       ignore(check_stmt globs fname_map fdecl stmt);

and check_for globs fname_map fdecl stmt =
       ignore(check_stmt globs fname_map fdecl stmt);

and check_else globs fname_map fdecl stmt =
       ignore(check_stmt globs fname_map fdecl stmt);;

let check_function globals fname_map global_st fdecl =
       ignore(List.iter check_not_void_formal fdecl.formals);
       ignore(List.iter check_not_void_local fdecl.locals);
       ignore(report_duplicate "function" ((List.map get_formal_id fdecl.formals) @ (List.map
get_local_id fdecl.locals) @ (List.map get_global_id globals)));
       ignore(check_function_return fdecl.fname fdecl.body fdecl.return_type);
       ignore(check_fbody globals fname_map fdecl fdecl.body);;

let check_functions global_st globals fdecls =
       let sast =
              let fname_map = fdecls_to_fname_map fdecls in
              ignore(report_duplicate "function" (List.map (fun fd -> fd.fname) fdecls));
              ignore(List.iter (check_function globals fname_map global_st) fdecls);
              let sfdecls = List.map (convert_fdecl_to_sfdecl globals fname_map) fdecls in
              (globals, sfdecls)
       in sast
```

A.3.10 plt/src/utils.ml

```
(*
 * COMS4115: CMAT Utilities
 *
 * Authors:
 *  - Marissa Ojeda
 *  - Daniel Rojas
 *  - Mike Berkowitz
 *  - Frank Cabada
```

```
 *)

(* Pretty Printer *)
open Ast
open Sast
open Parser

let save file string =
        let channel = open_out file in
        output_string channel string;
        close_out channel

let string_of_num = function
              Int_lit(x) -> string_of_int x
       | Float_lit(x) -> string_of_float x

(* Print data types *)
let string_of_primitive = function
       Int                          -> "int"
       | Float              -> "float"
       | Void               -> "void"
       | Bool               -> "bool"
       | String             -> "String"
       | Vector(p,i)  -> "vector(" ^ (string_of_num i) ^ ")"
       | Matrix(p,i,j) -> "matrix(" ^ (string_of_num i) ^ "," ^ (string_of_num j) ^ ")"

let rec print_brackets = function
                     1      -> "[]"
       |      a -> "[]" ^ print_brackets (a - 1)

let string_of_datatype = function
       Datatype(p)          -> (string_of_primitive p)

(* Print expressions *)
let string_of_op = function
       Add                          -> "+"
        |      Sub                  -> "-"
        |      Mult           -> "*"
        |      Div                  -> "/"
        |      Equal          -> "=="
        |      Neq                  -> "!="
        |      Less           -> "<"
        |      Leq                  -> "<="
        |      Greater        -> ">"
        |      Geq                  -> ">="
        |      And                  -> "and"
        |      Or                   -> "or"

let string_of_uop = function
              Not                   -> "not"
       | Inc                -> "++"
       | Dec                -> "--"
       | Neg                -> "-"
```

```
let rec string_of_bracket_expr = function
       []                                  -> ""
       |      head :: tail   -> "[" ^ (string_of_expr head) ^ "]" ^ (string_of_bracket_expr tail)

and string_of_expr = function
       Num_lit(i)                                  -> string_of_num i
       | Bool_lit(b)                         -> if b then "true" else "false"
       | String_lit(s)                             -> "\"" ^ (String.escaped s) ^ "\""
       | Id(s)                                     -> s
       | Binop(e1, o, e2)                    -> (string_of_expr e1) ^ " " ^ (string_of_op o) ^ " " ^
(string_of_expr e2)
       | Assign(e1, e2)                      -> (string_of_expr e1) ^ " = " ^ (string_of_expr e2)
       | Noexpr                                    -> ""
       | Call(f, el)                         -> f ^ "(" ^ String.concat ", " (List.map string_of_expr
el) ^ ")"
       | Unop(uop, e)                        -> (string_of_uop uop) ^ "(" ^ string_of_expr e ^ ")"
       | Null                                      -> "null"
       | Matrix_lit(el)                      -> "Matrix_lit"
       | Vector_lit(e)             -> "Vector lit"
       | Vector_access (s, i)                -> (s) ^ "[" ^ (string_of_expr i) ^ "]"
       | Matrix_access (s, i, j)     -> (s) ^ "[" ^ (string_of_expr i) ^ "," ^ (string_of_expr j) ^
"]"
       | Matrix_row (s, i)                   -> (s) ^ "[" ^ (string_of_expr i)^ ",:]"
       | Matrix_col (s, j)                   -> (s) ^ "[:," ^ (string_of_expr j) ^ "]"
       | Rows(s)                                   -> (s) ^ ":rows"
       | Cols(s)                                   -> (s) ^ ":cols"
       | Transpose(s)                              -> (s) ^ ":tr"
       | Len(s)                                    -> (s) ^ ":len"

let string_of_snum = function
              SInt_lit(x) -> string_of_int x
       |      SFloat_lit(x) -> string_of_float x

let rec string_of_bracket_sexpr = function
              []                                  -> ""
       |      head :: tail   -> "[" ^ (string_of_sexpr head) ^ "]" ^ (string_of_bracket_sexpr tail)

and string_of_sarray_primitive = function
              []                                  -> ""
       |   [last]                        -> (string_of_sexpr last)
       |      head :: tail   -> (string_of_sexpr head) ^ ", " ^ (string_of_sarray_primitive tail)

and string_of_sexpr = function
       SNum_lit(i)                                         -> string_of_snum i
       | SBool_lit(b)                                -> if b then "true" else "false"
       | SString_lit(s)                              -> "\"" ^ (String.escaped s) ^ "\""
       | SId(s, _)                                         -> s
       | SBinop(e1, o, e2, _)                        -> (string_of_sexpr e1) ^ " " ^ (string_of_op o) ^
" " ^ (string_of_sexpr e2)
       | SAssign(se1, se2, _)                        -> (string_of_sexpr se1) ^ " = " ^
(string_of_sexpr se2)
       | SNoexpr                                           -> ""
       | SCall(f, el, _)                             -> f ^ "(" ^ String.concat ", " (List.map
string_of_sexpr el) ^ ")"
```

```
        | SUnop(uop, e, _)                          -> (string_of_uop uop) ^ "(" ^ string_of_sexpr e ^
")"
        | SNull                                             -> "null"
        | SMatrix_lit (_, _)                -> "SMatrix_lit"
        | SVector_lit (_, _)            -> "SVector_lit"
        | SVector_access (s, i, _)          -> (s) ^ "[" ^ (string_of_sexpr i) ^ "]"
        | SMatrix_access (s, i, j, _)       -> (s) ^ "[" ^ (string_of_sexpr i) ^ "," ^
(string_of_sexpr j) ^ "]"
        | SMatrix_row (s, i, _)                     -> (s) ^ "[" ^ (string_of_sexpr i)^ ",:]"
        | SMatrix_col (s, j, _)                     -> (s) ^ "[:," ^ (string_of_sexpr j) ^ "]"
        | SCols(c)                                          -> "SCols"
        | SRows(r)                                          -> "SRows"
        | STranspose(s, _)                          -> "STranspose"
        | SLen(l)                                           -> "SLen"

let string_of_local_expr = function
                Noexpr -> ""
        |       e          -> " = " ^ string_of_expr e

(* Print statements *)
let rec string_of_stmt indent =
        let indent_string = String.make indent '\t' in
        let get_stmt_string = function
                        Block(stmts)                    ->
                                indent_string ^ "{\n" ^
                                        String.concat "" (List.map (string_of_stmt (indent+1)) stmts) ^
                                indent_string ^ "}\n"

                |       Expr(expr)                              ->
                                indent_string ^ string_of_expr expr ^ ";\n";

                |       Return(expr)                    ->
                                indent_string ^ "return " ^ string_of_expr expr ^ ";\n";

                |       If(e, s, Block([Expr(Noexpr)]))         ->
                                indent_string ^ "if (" ^ string_of_expr e ^ ")\n" ^
                                        (string_of_stmt (indent+1) s)

                |       If(e, s1, s2)                   ->
                                indent_string ^ "if (" ^ string_of_expr e ^ ")\n" ^
                                        string_of_stmt (indent+1) s1 ^
                                indent_string ^ "else\n" ^
                                        string_of_stmt (indent+1) s2

                |       For(e1, e2, e3, s)              ->
                                indent_string ^ "for (" ^ string_of_expr e1  ^ " ; " ^ string_of_expr e2
^ " ; " ^ string_of_expr e3  ^ ")\n" ^
                                        string_of_stmt (indent) s

                |       While(e, s)                     ->
                                indent_string ^ "while (" ^ string_of_expr e ^ ")\n" ^
                                        string_of_stmt (indent) s

        in get_stmt_string
```

```
let string_of_local_sexpr = function
              SNoexpr       -> ""
       |      e                    -> " = " ^ string_of_sexpr e

let rec string_of_sstmt indent =
       let indent_string = String.make indent '\t' in
       let get_stmt_string = function

                     SBlock(stmts)               ->
                            indent_string ^ "{\n" ^
                                   String.concat "" (List.map (string_of_sstmt (indent+1)) stmts) ^
                            indent_string ^ "}\n"

              |      SExpr(expr)                        ->
                            indent_string ^ string_of_sexpr expr ^ ";\n";

              |      SReturn(expr)               ->
                            indent_string ^ "return " ^ string_of_sexpr expr ^ ";\n";

              |      SIf(e, s, SBlock([SExpr(SNoexpr)])) ->
                            indent_string ^ "if (" ^ string_of_sexpr e ^ ")\n" ^
                                   (string_of_sstmt (indent+1) s)

              |      SIf(e, s1, s2)                     ->
                            indent_string ^ "if (" ^ string_of_sexpr e ^ ")\n" ^
                                   string_of_sstmt (indent+1) s1 ^
                            indent_string ^ "else\n" ^
                                   string_of_sstmt (indent+1) s2

              |      SFor(e1, e2, e3, s)          ->
                            indent_string ^ "for (" ^ string_of_sexpr e1  ^ " ; " ^ string_of_sexpr
e2 ^ " ; " ^ string_of_sexpr e3  ^ ")\n" ^
                                   string_of_sstmt (indent) s

              |      SWhile(e, s)                ->
                            indent_string ^ "while (" ^ string_of_sexpr e ^ ")\n" ^
                                   string_of_sstmt (indent) s

       in get_stmt_string

(* Print Function *)

let string_of_formal = function
       Formal(d, s) -> (string_of_datatype d) ^ " " ^ s

let string_of_formal_name = function
       Formal(_, s) -> s

let string_of_local = function
       Local(d, s) -> (string_of_datatype d) ^ " " ^ s

let string_of_func_decl fdecl =
       "" ^ (string_of_datatype fdecl.return_type) ^ " " ^ (fdecl.fname) ^ " " ^
```

```
        (* Formals *)
        "(" ^ String.concat "," (List.map string_of_formal fdecl.formals) ^ ") {\n" ^
        (* Locals *)
        String.concat "" (List.map string_of_local fdecl.locals) ^
        (* body *)
        String.concat "" (List.map (string_of_stmt 2) fdecl.body) ^
        "\t}\n\n"

let string_of_vdecl = function
        (d, s) -> (string_of_datatype d) ^ " " ^ s ^ ";\n"

(* Print whole program *)
let string_of_program = function
        (vars, fdecls) ->
                String.concat "" (List.map string_of_vdecl vars) ^ "\n" ^
                String.concat "\n" (List.map string_of_func_decl fdecls)
```

A.4 plt/test

A.4.1 plt/test/compiler/fail

A.4.1.1 plt/test/compiler/fail/_dupe_global_local.test

```
float x;

int main() {
  int x;
  print_string("Hello world!");
  return 0;
}
```

A.4.1.2 plt/test/compiler/fail/_dupe_global_local.out

```
Fatal error: exception Exceptions.DuplicateFunc("x")
```

A.4.1.3 plt/test/compiler/fail/_duplicate_global.test

```
int a;
float a;

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

A.4.1.4 plt/test/compiler/fail/_duplicate_global.out

```
Fatal error: exception Exceptions.DuplicateGlobal("a")
```

A.4.1.5 plt/test/compiler/fail/_function_nonexistent.test

```
int main() {
  foo();
  return 0;
```

```
}
```

A.4.1.6 plt/test/compiler/fail/_function_nonexistent.out

```
Fatal error: exception Exceptions.FunctionNotFound("foo")
```

A.4.1.7 plt/test/compiler/fail/_illegal_float_equality.test

```
int main() {
  float a;
  float b;
  a == b;
  return 0;
}
```

A.4.1.8 plt/test/compiler/fail/_illegal_float_equality.out

```
Fatal error: exception Exceptions.InvalidBinopExpression("Cannot use equality for floats")
```

A.4.1.9 plt/test/compiler/fail/_incorrect_function_args.test

```
int main() {
  int a;
  a = sum(1,2,3);
  return 0;
}

int sum(int x, int y) {
  return x+y;
}
```

A.4.1.10 plt/test/compiler/fail/_incorrect_function_args.out

```
Fatal error: exception Exceptions.IncorrectNumberOfArguments("sum", 3, 2)
```

A.4.1.11 plt/test/compiler/fail/_matrix_equality.test

```
int main() {
  matrix int [2,3] m1;
  matrix int [2,3] m2;
  m1 == m2;
  return 0;
}
```

A.4.1.12 plt/test/compiler/fail/_matrix_equality.out

```
Fatal error: exception Exceptions.InvalidBinopExpression("Can only use equality operators with ints
and Strings")
```

A.4.1.13 plt/test/compiler/fail/_matrix_mismatch_add.test

```
int main() {
  matrix int [3,2] m1;
  matrix int [2,3] m2;
  m1 = [1,2;3,4;5,6];
  m2 = [1,2,3;4,5,6];
```

```
  m2 = m1 + m2;
  return 0;
}
```

A.4.1.14 plt/test/compiler/fail/_matrix_mismatch_add.out

```
Fatal error: exception Exceptions.MismatchedMatricesForAddSub("Matrices must be same type and
dimensions for +/-")
```

A.4.1.15 plt/test/compiler/fail/_no_return.test

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

int foo() {

}
```

A.4.1.16 plt/test/compiler/fail/_no_return.out

```
Fatal error: exception Exceptions.AllNonVoidFunctionsMustEndWithReturn("foo")
```

A.4.1.17 plt/test/compiler/fail/_return_mismatch.test

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

int foo() {
  return "hello";
}
```

A.4.1.18 plt/test/compiler/fail/_return_mismatch.out

```
Fatal error: exception Exceptions.ReturnTypeMismatch("String", "int")
```

A.4.1.19 plt/test/compiler/fail/_void_formal.test

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

int foo(void x) {
  return 1;
}
```

A.4.1.20 plt/test/compiler/fail/_void_formal.out

```
Fatal error: exception Exceptions.VoidFunctionFormal("x")
```

A.4.1.21 plt/test/compiler/fail/_void_global.test

```
void a;
```

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

A.4.1.22 plt/test/compiler/fail/_void_global.out

```
Fatal error: exception Exceptions.VoidGlobal("a")
```

A.4.1.23 plt/test/compiler/fail/_void_local.test

```
int main() {
  void x;
  return 0;
}
```

A.4.1.24 plt/test/compiler/fail/_void_local.out

```
Fatal error: exception Exceptions.VoidFunctionLocal("x")
```

A.4.1.25 plt/test/compiler/fail/_void_return.test

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

void foo() {
  return 0;
}
```

A.4.1.26 plt/test/compiler/fail/_void_return.out

```
Fatal error: exception Exceptions.AllVoidFunctionsMustNotReturn("foo")
```

A.4.2 plt/test/compiler/pass

A.4.2.1 plt/test/compiler/pass/_arithmetic_binops.test

```
int main() {
    int i;
    int j;
    int k;
    float f;
    float g;
    float h;
    i = 2;
    j = 4;
    f = 2.2;
    g = 3.3;
    k = i+j;
    print_int(k);
    k = i-j;
    print_int(k);
    k = i*j;
```

```
    print_int(k);
    k = j/i;
    print_int(k);
    h = g+f;
    print_float(h);
    h = g-f;
    print_float(h);
    h = g*f;
    print_float(h);
    h = g/f;
    print_float(h);
    return 0;
}
```

A.4.2.2 plt/test/compiler/pass/_arithmetic_binops.out

```
6       -2      8       2       5.500000        1.100000        7.260000        1.500000
```

A.4.2.3 plt/test/compiler/pass/_assign.test

```
int g;
int main() {
    int i;
    float f;
    vector float [5] v;
    matrix int [4, 3] m;
    m[0,0] = 1;
    m[0,1] = 2;
    m[0,2] = 3;
    m[1,0] = 4;
    m[1,1] = 5;
    m[1,2] = 6;
    m[2,0] = 7;
    m[2,1] = 8;
    m[2,2] = 9;
    v[0] = 0.5; v[1] = 1.5; v[2] = 2.5; v[3] = 3.5; v[4] = 4.5;
    g=2;
    i = 5;
    f = 42.42;
    print_int(g);
    print_int(i);
    print_float(f);
    print_int(m[0,0]);
    print_int(m[0,1]);
    print_int(m[0,2]);
    print_int(m[1,0]);
    print_int(m[1,1]);
    print_int(m[1,2]);
    print_int(m[2,0]);
    print_int(m[2,1]);
    print_int(m[2,2]);
    print_float(v[0]);
    print_float(v[1]);
    print_float(v[2]);
    print_float(v[3]);
```

```
    print_float(v[4]);
    print_int(m:rows);
    print_int(m:cols);
    print_int(v:len);
    return 0;
}
```

A.4.2.4 plt/test/compiler/pass/_assign.out

```
2       5       42.420000       1       2       3       4       5       6       7       8       9
        0.500000        1.500000        2.500000        3.500000        4.500000        4       3       5
```

A.4.2.5 plt/test/compiler/pass/_control_flow.test

```
int main() {
    int i;
    int j;
    i=1;
    while(i < 4) {
        for(j = 0; j < 3; ++j) {
            if(i == 1) { print_string("i=1"); }
            else if(i == 2) { print_string("i=2"); }
            else { print_string("i=3"); }
        }
        i = i+1;
    }
    return 0;
}
```

A.4.2.6 plt/test/compiler/pass/_control_flow.out

```
i=1
i=1
i=1
i=2
i=2
i=2
i=3
i=3
i=3
```

A.4.2.7 plt/test/compiler/pass/_func_call.test

```
int main() {
    int i;
    i=4;
    print_int(foo(i));
    return 0;
}

int foo(int j) {
    int i;
    i = j + 6;
    return i;
}
```

A.4.2.8 plt/test/compiler/pass/_func_call.out

```
10
```

A.4.2.9 plt/test/compiler/pass/_inc.cmat

```
void foo() {
    print_string("Hello!");
}
```

A.4.2.10 plt/test/compiler/pass/_include.test

```
#include <pass/inc.cmat>;

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

A.4.2.11 plt/test/compiler/pass/_include.out

```
Hello!
```

A.4.2.12 plt/test/compiler/pass/_matrices.test

```
#include <../../stdlib.cmat>;

int main() {
    matrix int [6,3] m63;
    matrix int [6,3] m63b;
    matrix int [3,5] m35;
    matrix int [6,5] m65;
    m63 = [1,0,0;
           0,1,0;
           0,0,1;
           0,0,0;
           0,0,0;
           0,0,0];
    m35 = [1,0,0,0,0;
           0,1,0,0,0;
           0,0,1,0,0];
   m63b = [1,0,0;
           0,1,0;
           0,0,1;
           0,0,0;
           0,0,0;
           0,0,0];
    printi63(m63);
    printi63(m63b);
    m63b = m63b+m63;
    printi63(m63b);
    m63b = m63b-m63;
    printi63(m63b);
    m65 = m63*m35;
    printi65(m65);
```

```
    print_int(m65:rows);
    print_int(m65:cols);
    print_string("");
    printi56(m65:tr);
    return 0;
}
```

A.4.2.13 plt/test/compiler/pass/_matrices.out

```
1       0       0
0       1       0
0       0       1
0       0       0
0       0       0
0       0       0
1       0       0
0       1       0
0       0       1
0       0       0
0       0       0
0       0       0
2       0       0
0       2       0
0       0       2
0       0       0
0       0       0
0       0       0
1       0       0
0       1       0
0       0       1
0       0       0
0       0       0
0       0       0
1       0       0       0       0
0       1       0       0       0
0       0       1       0       0
0       0       0       0       0
0       0       0       0       0
0       0       0       0       0
6       5
1       0       0       0       0       0
0       1       0       0       0       0
0       0       1       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0
```

A.4.2.14 plt/test/compiler/pass/_mat_vec.test

```
#include <../../stdlib.cmat>;

int main() {
    int i;
    int j;
    float f;
```

```
matrix int [2,3] mi;
matrix int [2,3] mj;
matrix float [3,5] me;
matrix float [3,5] mf;
vector int [2] vi;
vector float [2] vf;
i=1;
j=2;
f=5.5;
vi[0] = 1;
vi[1] = 2;
vf[0] = 1.1;
vf[1] = 2.2;
mi = [1,2,3;4,5,6];
mj = [1,2,3;4,5,6];
me = [1.1,2.2,3.3,4.4,5.5;1.1,2.2,3.3,4.4,5.5;1.1,2.2,3.3,4.4,5.5];
mf = [1.1,2.2,3.3,4.4,5.5;1.1,2.2,3.3,4.4,5.5;1.1,2.2,3.3,4.4,5.5];

print_string("vi: ");
printi2(vi);
print_string("vi+vi: ");
printi2(vi+vi);
print_string("vi-vi: ");
printi2(vi-vi);
print_string("j*vi: ");
printi2(j*vi);
print_string("vf: ");
printf2(vf);
print_string("vf+vf: ");
printf2(vf+vf);
print_string("vf-vf: ");
printf2(vf-vf);
print_string("f*vf: ");
printf2(f*vf);
print_string("mi:");
printi23(mi);
print_string("mj:");
printi23(mj);
print_string("mi+mj:");
printi23(mi+mj);
print_string("mi-mj:");
printi23(mi-mj);
print_string("j*mi:");
printi23(j*mi);
print_string("me:");
printf35(me);
print_string("mf:");
printf35(mf);
print_string("mf+me:");
printf35(mf+me);
print_string("mf-me:");
printf35(mf-me);
print_string("f*mf:");
printf35(f*mf);
```

```
    return 0;
}
```

A.4.2.15 plt/test/compiler/pass/_mat_vec.out

```
vi:
1       2
vi+vi:
2       4
vi-vi:
0       0
j*vi:
2       4
vf:
1.100000        2.200000
vf+vf:
2.199999        4.400000
vf-vf:
0.000000        0.000000
f*vf:
6.049998        12.100000
mi:
1       2       3
4       5       6
mj:
1       2       3
4       5       6
mi+mj:
2       4       6
8       10      12
mi-mj:
0       0       0
0       0       0
j*mi:
2       4       6
8       10      12
me:
1.100000        2.200000        3.300000        4.400000        5.500000
1.100000        2.200000        3.300000        4.400000        5.500000
1.100000        2.200000        3.300000        4.400000        5.500000
mf:
1.100000        2.200000        3.300000        4.400000        5.500000
1.100000        2.200000        3.300000        4.400000        5.500000
1.100000        2.200000        3.300000        4.400000        5.500000
mf+me:
2.200000        4.400000        6.600000        8.800000        11.000000
2.200000        4.400000        6.600000        8.800000        11.000000
2.200000        4.400000        6.600000        8.800000        11.000000
mf-me:
0.000000        0.000000        0.000000        0.000000        0.000000
0.000000        0.000000        0.000000        0.000000        0.000000
0.000000        0.000000        0.000000        0.000000        0.000000
f*mf:
6.050000        12.100000       18.150000       24.200000       30.250000
```

```
6.050000        12.100000       18.150000       24.200000       30.250000
6.050000        12.100000       18.150000       24.200000       30.250000
```

A.4.2.16 plt/test/compiler/pass/_return.test

```
int main() {
    int i;
    float f;
    bool t;
    bool fa;
    i=0;
    f = foo();
    i = bar();
    t = foobar();
    fa = false;
    if(t) {
        print_int(i);
        print_float(f);
    }
    if(fa) {}
    else print_string("Hello!");
    return i;
}

float foo() {
    return 1.1;
}

int bar() {
    return 42;
}

bool foobar() {
    return true;
}
```

A.4.2.17 plt/test/compiler/pass/_return.out

```
42      1.100000        Hello!
```

A.4.2.18 plt/test/compiler/pass/_vectors.test

```
#include <../../stdlib.cmat>;

int main() {
    vector int [8] v1;
    vector int [8] v2;
    v1 = | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |;
    v2 = | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |;
    printi8(v1);
    printi8(v2);
    print_int(doti8(v1,v2));
    print_string("");
    v1 = v1+v2;
```

```
    printi8(v1);
    v1 = v1-v2;
    return 0;
}
```

A.4.2.19 plt/test/compiler/pass/_vectors.out

```
1       1       1       1       1       1       1       1
1       1       1       1       1       1       1       1
8
2       2       2       2       2       2       2       2
```

A.4.3 plt/test/compiler/scripts/build.sh

```
#!/bin/bash

cp ../../src/scanner.mll ./scanner.mll
cp ../../src/parser.mly ./parser.mly
cp ../../src/ast.ml ./ast.ml
cp ../../src/sast.ml ./sast.ml
cp ../../src/semant.ml ./semant.ml
cp ../../src/exceptions.ml ./exceptions.ml
cp ../../src/utils.ml ./utils.ml
cp ../../src/codegen.ml ./codegen.ml
cp ../../src/prep.ml ./prep.ml
cp ../../src/cmat.ml ./cmat.ml

ocamlbuild -j 0 -r -use-ocamlfind -pkgs
str,llvm,llvm.analysis,llvm.bitwriter,llvm.bitreader,llvm.linker,llvm.target cmat.native
```

A.4.4 plt/test/compiler/scripts/clean.sh

```
#!/bin/bash

rm -rf _build *.cmo *.cmi pass/*.res fail/*.res fail/*.ll *.ml* cmat.native build.log pass/*.ll
```

A.4.5 plt/test/compiler/scripts/test.sh

```
#! /bin/bash

./cmat.native -c pass/_assign.test pass/_assign.ll
lli pass/_assign.ll > pass/_assign.res
diff pass/_assign.out pass/_assign.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|      COMPILER: ASSIGN TEST PASSED     |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|      COMPILER: ASSIGN TEST FAILED     |"
```

```
  echo "----------------------------------------"
fi

./cmat.native -c pass/_include.test pass/_include.ll
lli pass/_include.ll > pass/_include.res
diff pass/_include.out pass/_include.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|     COMPILER: INCLUDE TEST PASSED     |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|     COMPILER: INCLUDE TEST FAILED     |"
  echo "----------------------------------------"
fi

./cmat.native -c pass/_func_call.test pass/_func_call.ll
lli pass/_func_call.ll > pass/_func_call.res
diff pass/_func_call.out pass/_func_call.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|    COMPILER: FUNC CALL TEST PASSED    |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|    COMPILER: FUNC CALL TEST FAILED    |"
  echo "----------------------------------------"
fi

./cmat.native -c pass/_arithmetic_binops.test pass/_arithmetic_binops.ll
lli pass/_arithmetic_binops.ll > pass/_arithmetic_binops.res
diff pass/_arithmetic_binops.out pass/_arithmetic_binops.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|  COMPILER: ARITH BINOPS TEST PASSED   |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|  COMPILER: ARITH BINOPS TEST FAILED   |"
  echo "----------------------------------------"
fi

./cmat.native -c pass/_return.test pass/_return.ll
lli pass/_return.ll > pass/_return.res
diff pass/_return.out pass/_return.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
```

```
  echo "----------------------------------------"
  echo "|     COMPILER: RETURN TEST PASSED      |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|     COMPILER: RETURN TEST FAILED      |"
  echo "----------------------------------------"
fi

./cmat.native -c pass/_control_flow.test pass/_control_flow.ll
lli pass/_control_flow.ll > pass/_control_flow.res
diff pass/_control_flow.out pass/_control_flow.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|  COMPILER: CONTROL FLOW TEST PASSED   |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|  COMPILER: CONTROL FLOW TEST FAILED   |"
  echo "----------------------------------------"
fi

./cmat.native -c pass/_matrices.test pass/_matrices.ll
lli pass/_matrices.ll > pass/_matrices.res
diff pass/_matrices.out pass/_matrices.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|     COMPILER: MATRIX TEST PASSED      |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|     COMPILER: MATRIX TEST FAILED      |"
  echo "----------------------------------------"
fi

./cmat.native -c pass/_vectors.test pass/_vectors.ll
lli pass/_vectors.ll > pass/_vectors.res
diff pass/_vectors.out pass/_vectors.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|     COMPILER: VECTOR TEST PASSED      |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|     COMPILER: VECTOR TEST FAILED      |"
  echo "----------------------------------------"
fi
```

```
./cmat.native -c pass/_mat_vec.test pass/_mat_vec.ll
lli pass/_mat_vec.ll > pass/_mat_vec.res
diff pass/_mat_vec.out pass/_mat_vec.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|  COMPILER: MATRIX/VECTOR TEST PASSED  |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|  COMPILER: MATRIX/VECTOR TEST FAILED  |"
  echo "----------------------------------------"
fi

./cmat.native -c fail/_illegal_float_equality.test fail/_illegal_float_equality.ll >&
fail/_illegal_float_equality.res
diff fail/_illegal_float_equality.out fail/_illegal_float_equality.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "| COMPILER: FLOAT EQUALITY TEST PASSED  |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "| COMPILER: FLOAT EQUALITY TEST FAILED  |"
  echo "----------------------------------------"
fi

./cmat.native -c fail/_matrix_equality.test fail/_matrix_equality.ll >& fail/_matrix_equality.res
diff fail/_matrix_equality.out fail/_matrix_equality.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "| COMPILER: MATRIX EQUALITY TEST PASSED |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "| COMPILER: MATRIX EQUALITY TEST FAILED |"
  echo "----------------------------------------"
fi

./cmat.native -c fail/_duplicate_global.test fail/_duplicate_global.ll >& fail/_duplicate_global.res
diff fail/_duplicate_global.out fail/_duplicate_global.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "| COMPILER: DUPLICATE GLOBAL TEST PASSED |"
  echo "----------------------------------------"
else
```

```
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "| COMPILER: DUPLICATE GLOBAL TEST FAILED |"
  echo "----------------------------------------"
fi

./cmat.native -c fail/_function_nonexistent.test fail/_function_nonexistent.ll >&
fail/_function_nonexistent.res
diff fail/_function_nonexistent.out fail/_function_nonexistent.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "| COMPILER:MISSING FUNCTION TEST PASSED |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "| COMPILER:MISSING FUNCTION TEST FAILED |"
  echo "----------------------------------------"
fi

./cmat.native -c fail/_incorrect_function_args.test fail/_incorrect_function_args.ll >&
fail/_incorrect_function_args.res
diff fail/_incorrect_function_args.out fail/_incorrect_function_args.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|  COMPILER: FUNCTION ARGS TEST PASSED  |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|  COMPILER: FUNCTION ARGS TEST FAILED  |"
  echo "----------------------------------------"
fi

./cmat.native -c fail/_matrix_mismatch_add.test fail/_matrix_mismatch_add.ll >&
fail/_matrix_mismatch_add.res
diff fail/_matrix_mismatch_add.out fail/_matrix_mismatch_add.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "| COMPILER: MATRIX MISMATCH TEST PASSED |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "| COMPILER: MATRIX MISMATCH TEST FAILED |"
  echo "----------------------------------------"
fi

./cmat.native -c fail/_no_return.test fail/_no_return.ll >& fail/_no_return.res
diff fail/_no_return.out fail/_no_return.res > /dev/null
```

```
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "| COMPILER: MISSING RETURN TEST PASSED  |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "| COMPILER: MISSING RETURN TEST FAILED  |"
  echo "----------------------------------------"
fi

./cmat.native -c fail/_return_mismatch.test fail/_return_mismatch.ll >& fail/_return_mismatch.res
diff fail/_return_mismatch.out fail/_return_mismatch.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "| COMPILER: RETURN MISMATCH TEST PASSED |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "| COMPILER: RETURN MISMATCH TEST FAILED |"
  echo "----------------------------------------"
fi

./cmat.native -c fail/_void_formal.test fail/_void_formal.ll >& fail/_void_formal.res
diff fail/_void_formal.out fail/_void_formal.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|   COMPILER: VOID FORMAL TEST PASSED   |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|   COMPILER: VOID FORMAL TEST FAILED   |"
  echo "----------------------------------------"
fi

./cmat.native -c fail/_void_global.test fail/_void_global.ll >& fail/_void_global.res
diff fail/_void_global.out fail/_void_global.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|   COMPILER: VOID GLOBAL TEST PASSED   |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|   COMPILER: VOID GLOBAL TEST FAILED   |"
  echo "----------------------------------------"
fi
```

```
./cmat.native -c fail/_void_local.test fail/_void_local.ll >& fail/_void_local.res
diff fail/_void_local.out fail/_void_local.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|   COMPILER: VOID LOCAL TEST PASSED    |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|   COMPILER: VOID LOCAL TEST FAILED    |"
  echo "----------------------------------------"
fi

./cmat.native -c fail/_void_return.test fail/_void_return.ll >& fail/_void_return.res
diff fail/_void_return.out fail/_void_return.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|   COMPILER: VOID RETURN TEST PASSED   |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|   COMPILER: VOID RETURN TEST FAILED   |"
  echo "----------------------------------------"
fi

./cmat.native -c fail/_dupe_global_local.test fail/_dupe_global_local.ll >&
fail/_dupe_global_local.res
diff fail/_dupe_global_local.out fail/_dupe_global_local.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|COMPILER: DUPE GLOBAL LOCAL TEST PASSED|"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|COMPILER: DUPE GLOBAL LOCAL TEST FAILED|"
  echo "----------------------------------------"
fi
```

A.4.6 plt/test/parser/fail

A.4.6.1 plt/test/parser/fail/_illegal_binop.test

```
NUM_LIT PLUS TIMES NUM_LIT SEMI
```

A.4.6.2 plt/test/parser/fail/_illegal_binop.out

```
REJECT
```

A.4.6.3 plt/test/parser/fail/_internal_fdecl.test

```
INT ID LPAREN RPAREN LBRACE INT ID LPAREN RPAREN LBRACE INT ID SEMI RBRACE RBRACE
```

A.4.6.4 plt/test/parser/fail/_internal_fdecl.out

```
REJECT
```

A.4.6.5 plt/test/parser/fail/_malformed_fdecl.test

```
INT ID LPAREN RPAREN LBRACE RETURN NUM_LIT SEMI INT ID SEMI RBRACE
```

A.4.6.6 plt/test/parser/fail/_malformed_fdecl.out

```
REJECT
```

A.4.6.7 plt/test/parser/fail/_malformed_matrix_decl.test

```
MATRIX INT LBRACKET RBRACKET ID SEMI
```

A.4.6.8 plt/test/parser/fail/_malformed_matrix_decl.out

```
REJECT
```

A.4.6.9 plt/test/parser/fail/_postfix_unop.test

```
INT ID LPAREN RPAREN LBRACE INT ID SEMI ID PLUS PLUS SEMI RBRACE
```

A.4.6.10 plt/test/parser/fail/_postfix_unop.out

```
REJECT
```

A.4.6.11 plt/test/parser/fail/_vdecl_without_semi.test

```
INT ID
```

A.4.6.12 plt/test/parser/fail/_vdecl_without_semi.out

```
REJECT
```

A.4.7 plt/test/parser/pass

A.4.7.1 plt/test/parser/pass/_base_parser.test

```
INT ID LPAREN RPAREN LBRACE RBRACE
```

A.4.7.2 plt/test/parser/pass/_base_parser.out

```
ACCEPT
[program:
  [decls:
    [decls:]
    [fdecl:
      [datatype: [primitives: INT]]
      ID
      LPAREN
```

```
      [formals_opt:]
      RPAREN
      LBRACE
      [vdecl_list:]
      [stmt_list:]
      RBRACE
    ]
  ]
  EOF
]
```

A.4.7.3 plt/test/parser/pass/_expr.test

```
INT ID LPAREN RPAREN LBRACE NUM_LIT SEMI STRING_LIT SEMI TRUE SEMI FALSE SEMI NULL SEMI ID SEMI
NUM_LIT PLUS NUM_LIT SEMI NUM_LIT MINUS NUM_LIT SEMI NUM_LIT TIMES NUM_LIT SEMI NUM_LIT DIVIDE
NUM_LIT SEMI NUM_LIT EQ NUM_LIT SEMI NUM_LIT NEQ NUM_LIT SEMI NUM_LIT LEQ NUM_LIT SEMI NUM_LIT GEQ
NUM_LIT SEMI NUM_LIT AND NUM_LIT SEMI NUM_LIT OR NUM_LIT SEMI MINUS NUM_LIT SEMI NOT NUM_LIT SEMI INC
NUM_LIT SEMI DEC NUM_LIT SEMI ID ASSIGN NUM_LIT SEMI LPAREN NUM_LIT RPAREN SEMI ID LPAREN RPAREN SEMI
LBRACKET NUM_LIT COMMA NUM_LIT SEMI NUM_LIT COMMA NUM_LIT RBRACKET SEMI ID LBRACKET NUM_LIT COMMA
NUM_LIT RBRACKET SEMI ID LBRACKET NUM_LIT COMMA COLON RBRACKET SEMI ID LBRACKET COLON COMMA NUM_LIT
RBRACKET SEMI RBRACE
```

A.4.7.4 plt/test/parser/pass/_expr.out

```
ACCEPT
[program:
  [decls:
    [decls:]
    [fdecl:
      [datatype: [primitives: INT]]
      ID
      LPAREN
      [formals_opt:]
      RPAREN
      LBRACE
      [vdecl_list:]
      [stmt_list:
        [stmt_list:
          [stmt_list:
            [stmt_list:
              [stmt_list:
                [stmt_list:
                  [stmt_list:
                    [stmt_list:
                      [stmt_list:
                        [stmt_list:
                          [stmt_list:
                            [stmt_list:
                              [stmt_list:
                                [stmt_list:
                                  [stmt_list:
                                    [stmt_list:
                                      [stmt_list:
                                        [stmt_list:
                                          [stmt_list:
```

```
[stmt_list:
  [stmt_list:
    [stmt_list:
      [stmt_list:
        [stmt_list:
          [stmt_list:
            [stmt_list:
              [stmt_list:
                [stmt_list:]
                [stmt:
                  [expr: NUM_LIT]
                  SEMI
                ]
              ]
              [stmt:
                [expr: STRING_LIT]
                SEMI
              ]
            ]
            [stmt: [expr: TRUE] SEMI]
          ]
          [stmt: [expr: FALSE] SEMI]
        ]
        [stmt: [expr: NULL] SEMI]
      ]
      [stmt: [expr: ID] SEMI]
    ]
    [stmt:
      [expr:
        [expr: NUM_LIT]
        PLUS
        [expr: NUM_LIT]
      ]
      SEMI
    ]
  ]
  [stmt:
    [expr:
      [expr: NUM_LIT]
      MINUS
      [expr: NUM_LIT]
    ]
    SEMI
  ]
]
[stmt:
  [expr:
    [expr: NUM_LIT]
    TIMES
    [expr: NUM_LIT]
  ]
  SEMI
]
]
```

```
                    [stmt:
                      [expr:
                        [expr: NUM_LIT]
                        DIVIDE
                        [expr: NUM_LIT]
                      ]
                      SEMI
                    ]
                  ]
                  [stmt:
                    [expr:
                      [expr: NUM_LIT]
                      EQ
                      [expr: NUM_LIT]
                    ]
                    SEMI
                  ]
                ]
                [stmt:
                  [expr:
                    [expr: NUM_LIT]
                    NEQ
                    [expr: NUM_LIT]
                  ]
                  SEMI
                ]
              ]
              [stmt:
                [expr: [expr: NUM_LIT] LEQ [expr: NUM_LIT]]
                SEMI
              ]
            ]
            [stmt:
              [expr: [expr: NUM_LIT] GEQ [expr: NUM_LIT]]
              SEMI
            ]
          ]
          [stmt:
            [expr: [expr: NUM_LIT] AND [expr: NUM_LIT]]
            SEMI
          ]
        ]
        [stmt:
          [expr: [expr: NUM_LIT] OR [expr: NUM_LIT]]
          SEMI
        ]
      ]
      [stmt: [expr: MINUS [expr: NUM_LIT]] SEMI]
    ]
    [stmt: [expr: NOT [expr: NUM_LIT]] SEMI]
  ]
  [stmt: [expr: INC [expr: NUM_LIT]] SEMI]
]
[stmt: [expr: DEC [expr: NUM_LIT]] SEMI]
```

```
                ]
                [stmt: [expr: [expr: ID] ASSIGN [expr: NUM_LIT]] SEMI]
              ]
              [stmt: [expr: LPAREN [expr: NUM_LIT] RPAREN] SEMI]
            ]
            [stmt: [expr: ID LPAREN [actuals_opt:] RPAREN] SEMI]
          ]
          [stmt:
            [expr:
              LBRACKET
              [mat_lit:
                [mat_lit:
                  [lit_list: [lit_list: [lit: NUM_LIT]] COMMA [lit: NUM_LIT]]
                ]
                SEMI
                [lit_list: [lit_list: [lit: NUM_LIT]] COMMA [lit: NUM_LIT]]
              ]
              RBRACKET
            ]
            SEMI
          ]
        ]
        [stmt:
          [expr:
            ID
            LBRACKET
            [expr: NUM_LIT]
            COMMA
            [expr: NUM_LIT]
            RBRACKET
          ]
          SEMI
        ]
      ]
      [stmt:
        [expr: ID LBRACKET [expr: NUM_LIT] COMMA COLON RBRACKET]
        SEMI
      ]
    ]
    [stmt:
      [expr: ID LBRACKET COLON COMMA [expr: NUM_LIT] RBRACKET]
      SEMI
    ]
  ]
  RBRACE
 ]
]
EOF
]
```

A.4.7.5 plt/test/parser/pass/_formal_opts.test

```
INT ID LPAREN RPAREN LBRACE RETURN SEMI RBRACE FLOAT ID LPAREN INT ID RPAREN LBRACE RBRACE
```

A.4.7.6 plt/test/parser/pass/_format_opts.out

```
ACCEPT
[program:
  [decls:
    [decls:
      [decls:]
      [fdecl:
        [datatype: [primitives: INT]]
        ID
        LPAREN
        [formals_opt:]
        RPAREN
        LBRACE
        [vdecl_list:]
        [stmt_list: [stmt_list:] [stmt: RETURN SEMI]]
        RBRACE
      ]
    ]
    [fdecl:
      [datatype: [primitives: FLOAT]]
      ID
      LPAREN
      [formals_opt: [formal_list: [datatype: [primitives: INT]] ID]]
      RPAREN
      LBRACE
      [vdecl_list:]
      [stmt_list:]
      RBRACE
    ]
  ]
  EOF
]
```

A.4.7.7 plt/test/parser/pass/_main_with_assign.test

```
INT ID LPAREN RPAREN LBRACE INT ID SEMI ID ASSIGN NUM_LIT SEMI RETURN NUM_LIT SEMI RBRACE
```

A.4.7.8 plt/test/parser/pass/_main_with_assign.out

```
ACCEPT
[program:
  [decls:
    [decls:]
    [fdecl:
      [datatype: [primitives: INT]]
      ID
      LPAREN
      [formals_opt:]
      RPAREN
      LBRACE
      [vdecl_list:
        [vdecl_list:]
        [vdecl: [datatype: [primitives: INT]] ID SEMI]
      ]
```

```
      [stmt_list:
        [stmt_list:
          [stmt_list:]
          [stmt: [expr: [expr: ID] ASSIGN [expr: NUM_LIT]] SEMI]
        ]
        [stmt: RETURN [expr: NUM_LIT] SEMI]
      ]
      RBRACE
    ]
  ]
  EOF
]
```

A.4.7.9 plt/test/parser/pass/_main_with_fdecl.test

```
INT ID LPAREN RPAREN LBRACE RETURN SEMI RBRACE VOID ID LPAREN RPAREN LBRACE RETURN SEMI RBRACE
```

A.4.7.10 plt/test/parser/pass/_main_with_fdecl.out

```
ACCEPT
[program:
  [decls:
    [decls:
      [decls:]
      [fdecl:
        [datatype: [primitives: INT]]
        ID
        LPAREN
        [formals_opt:]
        RPAREN
        LBRACE
        [vdecl_list:]
        [stmt_list: [stmt_list:] [stmt: RETURN SEMI]]
        RBRACE
      ]
    ]
    [fdecl:
      [datatype: [primitives: VOID]]
      ID
      LPAREN
      [formals_opt:]
      RPAREN
      LBRACE
      [vdecl_list:]
      [stmt_list: [stmt_list:] [stmt: RETURN SEMI]]
      RBRACE
    ]
  ]
  EOF
]
```

A.4.7.11 plt/test/parser/pass/_main_with_return.test

```
INT ID LPAREN RPAREN LBRACE RETURN SEMI RBRACE
```

A.4.7.12 plt/test/parser/pass/_main_with_return.out

```
ACCEPT
[program:
  [decls:
    [decls:]
    [fdecl:
      [datatype: [primitives: INT]]
      ID
      LPAREN
      [formals_opt:]
      RPAREN
      LBRACE
      [vdecl_list:]
      [stmt_list: [stmt_list:] [stmt: RETURN SEMI]]
      RBRACE
    ]
  ]
  EOF
]
```

A.4.7.13 plt/test/parser/pass/_primitive_decls.test

```
INT ID LPAREN RPAREN LBRACE INT ID SEMI BOOL ID SEMI FLOAT ID SEMI STRING ID SEMI VECTOR INT LBRACKET
NUM_LIT RBRACKET ID SEMI MATRIX INT LBRACKET NUM_LIT COMMA NUM_LIT RBRACKET ID SEMI RETURN NUM_LIT
SEMI RBRACE
```

A.4.7.14 plt/test/parser/pass/_primitive_decls.out

```
ACCEPT
[program:
  [decls:
    [decls:]
    [fdecl:
      [datatype: [primitives: INT]]
      ID
      LPAREN
      [formals_opt:]
      RPAREN
      LBRACE
      [vdecl_list:
        [vdecl_list:
          [vdecl_list:
            [vdecl_list:
              [vdecl_list:
                [vdecl_list:
                  [vdecl_list:]
                  [vdecl: [datatype: [primitives: INT]] ID SEMI]
                ]
                [vdecl: [datatype: [primitives: BOOL]] ID SEMI]
              ]
              [vdecl: [datatype: [primitives: FLOAT]] ID SEMI]
            ]
            [vdecl: [datatype: [primitives: STRING]] ID SEMI]
          ]
```

```
          [vdecl:
            [datatype:
              [primitives: VECTOR [primitives: INT] LBRACKET NUM_LIT RBRACKET]
            ]
            ID
            SEMI
          ]
        ]
        [vdecl:
          [datatype:
            [primitives:
              MATRIX
              [primitives: INT]
              LBRACKET
              NUM_LIT
              COMMA
              NUM_LIT
              RBRACKET
            ]
          ]
          ID
          SEMI
        ]
      ]
      [stmt_list: [stmt_list:] [stmt: RETURN [expr: NUM_LIT] SEMI]]
      RBRACE
    ]
  ]
  EOF
]
```

A.4.7.15 plt/test/parser/pass/_stmts.test

```
INT ID LPAREN RPAREN LBRACE NUM_LIT SEMI IF LPAREN NUM_LIT RPAREN LBRACE ID ASSIGN NUM_LIT SEMI FOR
LPAREN SEMI TRUE SEMI RPAREN LBRACE ID ASSIGN ID MINUS ID SEMI RBRACE WHILE LPAREN TRUE RPAREN LBRACE
ID ASSIGN ID PLUS ID SEMI RBRACE RBRACE RETURN NUM_LIT SEMI RBRACE
```

A.4.7.16 plt/test/parser/pass/_stmts.out

```
ACCEPT
[program:
  [decls:
    [decls:]
    [fdecl:
      [datatype: [primitives: INT]]
      ID
      LPAREN
      [formals_opt:]
      RPAREN
      LBRACE
      [vdecl_list:]
      [stmt_list:
        [stmt_list:
          [stmt_list: [stmt_list:] [stmt: [expr: NUM_LIT] SEMI]]
          [stmt:
```

```
IF
LPAREN
[expr: NUM_LIT]
RPAREN
[stmt:
  LBRACE
  [stmt_list:
    [stmt_list:
      [stmt_list:
        [stmt_list:]
        [stmt: [expr: [expr: ID] ASSIGN [expr: NUM_LIT]] SEMI]
      ]
      [stmt:
        FOR
        LPAREN
        [expr_opt:]
        SEMI
        [expr: TRUE]
        SEMI
        [expr_opt:]
        RPAREN
        [stmt:
          LBRACE
          [stmt_list:
            [stmt_list:]
            [stmt:
              [expr:
                [expr: ID]
                ASSIGN
                [expr: [expr: ID] MINUS [expr: ID]]
              ]
              SEMI
            ]
          ]
          RBRACE
        ]
      ]
    ]
    [stmt:
      WHILE
      LPAREN
      [expr: TRUE]
      RPAREN
      [stmt:
        LBRACE
        [stmt_list:
          [stmt_list:]
          [stmt:
            [expr:
              [expr: ID]
              ASSIGN
              [expr: [expr: ID] PLUS [expr: ID]]
            ]
            SEMI
```

```
                    ]
                  ]
                  RBRACE
                ]
              ]
              ]
              RBRACE
            ]
          ]
        ]
        [stmt: RETURN [expr: NUM_LIT] SEMI]
      ]
      RBRACE
    ]
  ]
  EOF
]
```

A.4.8 plt/test/parser/scripts/clean.sh

```
#!/bin/bash

rm -f *.cmo *.cmi pass/*.res fail/*.res parser.* scanner.* ast.* parserize
```

A.4.9 plt/test/parser/scripts/test.sh

```
cp ../../src/parser.mly ./parser.mly

cat pass/_base_parser.test | menhir --interpret --interpret-show-cst parser.mly >
pass/_base_parser.res
diff pass/_base_parser.out pass/_base_parser.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|       PARSER: FIRST TEST PASSED       |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|       PARSER: FIRST TEST FAILED       |"
  echo "----------------------------------------"
fi

cat pass/_main_with_return.test | menhir --interpret --interpret-show-cst parser.mly >
pass/_main_with_return.res
diff pass/_main_with_return.out pass/_main_with_return.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "| PARSER: MAIN WITH RETURN TEST PASSED  |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
```

```
  echo "| PARSER: MAIN WITH RETURN TEST FAILED  |"
  echo "----------------------------------------"
fi

cat pass/_main_with_return.test | menhir --interpret --interpret-show-cst parser.mly >
pass/_main_with_return.res
diff pass/_main_with_return.out pass/_main_with_return.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|  PARSER: MAIN WITH FDECL TEST PASSED  |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|  PARSER: MAIN WITH FDECL TEST FAILED  |"
  echo "----------------------------------------"
fi

cat pass/_main_with_assign.test | menhir --interpret --interpret-show-cst parser.mly >
pass/_main_with_assign.res
diff pass/_main_with_assign.out pass/_main_with_assign.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "| PARSER: MAIN WITH ASSIGN TEST PASSED  |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "| PARSER: MAIN WITH ASSIGN TEST FAILED  |"
  echo "----------------------------------------"
fi

cat pass/_primitive_decls.test | menhir --interpret --interpret-show-cst parser.mly >
pass/_primitive_decls.res
diff pass/_primitive_decls.out pass/_primitive_decls.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "| PARSER: PRIMITIVE VDECLS TEST PASSED  |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "| PARSER: PRIMITIVE VDECLS TEST FAILED  |"
  echo "----------------------------------------"
fi

cat pass/_formal_opts.test | menhir --interpret --interpret-show-cst parser.mly >
pass/_formal_opts.res
diff pass/_formal_opts.out pass/_formal_opts.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
```

```
  echo "----------------------------------------"
  echo "|    PARSER: FORMAL OPTS TEST PASSED    |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|    PARSER: FORMAL OPTS TEST FAILED    |"
  echo "----------------------------------------"
fi

cat pass/_stmts.test | menhir --interpret --interpret-show-cst parser.mly > pass/_stmts.res
diff pass/_stmts.out pass/_stmts.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|    PARSER: STATEMENTS TEST PASSED     |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|    PARSER: STATEMENTS TEST FAILED     |"
  echo "----------------------------------------"
fi

cat pass/_expr.test | menhir --interpret --interpret-show-cst parser.mly > pass/_expr.res
diff pass/_expr.out pass/_expr.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|   PARSER: EXPRESSIONS TEST PASSED     |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|   PARSER: EXPRESSIONS TEST FAILED     |"
  echo "----------------------------------------"
fi

cat fail/_vdecl_without_semi.test | menhir --interpret --interpret-show-cst parser.mly >
fail/_vdecl_without_semi.res
diff fail/_vdecl_without_semi.out fail/_vdecl_without_semi.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|  PARSER: VDECL W/O SEMI TEST PASSED   |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|  PARSER: VDECL W/O SEMI TEST FAILED   |"
  echo "----------------------------------------"
fi

cat fail/_illegal_binop.test | menhir --interpret --interpret-show-cst parser.mly >
```

```
fail/_illegal_binop.res
diff fail/_illegal_binop.out fail/_illegal_binop.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|   PARSER: ILLEGAL BINOP TEST PASSED   |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|   PARSER: ILLEGAL BINOP TEST FAILED   |"
  echo "----------------------------------------"
fi

cat fail/_internal_fdecl.test | menhir --interpret --interpret-show-cst parser.mly >
fail/_internal_fdecl.res
diff fail/_internal_fdecl.out fail/_internal_fdecl.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|   PARSER: INTERNAL FDECL TEST PASSED  |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|   PARSER: INTERNAL FDECL TEST FAILED  |"
  echo "----------------------------------------"
fi

cat fail/_malformed_fdecl.test | menhir --interpret --interpret-show-cst parser.mly >
fail/_malformed_fdecl.res
diff fail/_malformed_fdecl.out fail/_malformed_fdecl.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|  PARSER: MALFORMED FDECL TEST PASSED  |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|  PARSER: MALFORMED FDECL TEST FAILED  |"
  echo "----------------------------------------"
fi

cat fail/_malformed_matrix_decl.test | menhir --interpret --interpret-show-cst parser.mly >
fail/_malformed_matrix_decl.res
diff fail/_malformed_matrix_decl.out fail/_malformed_matrix_decl.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|  PARSER: BAD MATRIX DECL TEST PASSED  |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
```

```
  echo "----------------------------------------"
  echo "|  PARSER: BAD MATRIX DECL TEST FAILED  |"
  echo "----------------------------------------"
fi

cat fail/_postfix_unop.test | menhir --interpret --interpret-show-cst parser.mly >
fail/_postfix_unop.res
diff fail/_postfix_unop.out fail/_postfix_unop.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|   PARSER: POSTFIX UNOP TEST PASSED    |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|   PARSER: POSTFIX UNOP TEST FAILED    |"
  echo "----------------------------------------"
fi
```

A.4.10 plt/test/scanner/fail

A.4.10.1 plt/test/scanner/fail/_illegal_carrot.test

```
^
```

A.4.10.2 plt/test/scanner/fail/_illegal_carrot.out

```
Fatal error: exception Failure("illegal character ^")
```

A.4.10.3 plt/test/scanner/fail/_illegal_dollar.test

```
$
```

A.4.10.4 plt/test/scanner/fail/_illegal_dollar.out

```
Fatal error: exception Failure("illegal character $")
```

A.4.10.5 plt/test/scanner/fail/_illegal_percent.test

```
%
```

A.4.10.6 plt/test/scanner/fail/_illegal_percent.out

```
Fatal error: exception Failure("illegal character %")
```

A.4.10.7 plt/test/scanner/fail/_illegal_period.test

```
.
```

A.4.10.8 plt/test/scanner/fail/_illegal_period.out

```
Fatal error: exception Failure("illegal character .")
```

A.4.10.9 plt/test/scanner/fail/_illegal_pound.test

```
#
```

A.4.10.10 plt/test/scanner/fail/_illegal_pound.out

```
Fatal error: exception Failure("illegal character #")
```

A.4.10.11 plt/test/scanner/fail/_illegal_tilde.test

```
~
```

A.4.10.12 plt/test/scanner/fail/_illegal_tilde.out

```
Fatal error: exception Failure("illegal character ~")
```

A.4.11 plt/test/scanner/pass

A.4.11.1 plt/test/scanner/pass/_arithmetic.test

```
+ - * / = ++ --
```

A.4.11.2 plt/test/scanner/pass/_arithmetic.out

```
PLUS
MINUS
TIMES
DIVIDE
ASSIGN
INC
DEC
```

A.4.11.3 plt/test/scanner/pass/_assignment.test

```
int a = 4
```

A.4.11.4 plt/test/scanner/pass/_assignment.out

```
INT
ID
ASSIGN
NUM_LIT
```

A.4.11.5 plt/test/scanner/pass/_base_scanner.test

```
( ) { } [ ]
if else while for return main
== != < > <= >= && || !
+ - * / = ++ --
int float bool void null String true false
; , :
hi 99 "hi" 1.0
```

A.4.11.6 plt/test/scanner/pass/_base_scanner.out

```
LPAREN
```

```
RPAREN
LBRACE
RBRACE
LBRACKET
RBRACKET
IF
ELSE
WHILE
FOR
RETURN
ID
EQ
NEQ
LT
GT
LEQ
GEQ
AND
OR
NOT
PLUS
MINUS
TIMES
DIVIDE
ASSIGN
INC
DEC
INT
FLOAT
BOOL
VOID
NULL
STRING
TRUE
FALSE
SEMI
COMMA
COLON
ID
NUM_LIT
STRING_LIT
NUM_LIT
```

A.4.11.7 plt/test/scanner/pass/_comment.test

```
/*
   This is a comment
   "Comment"
   None of this should be tokenized.
   int num = 8;
 */
```

A.4.11.8 plt/test/scanner/pass/_comment.out

A.4.11.9 plt/test/scanner/pass/_conditionals.test

```
== != < > <= >= && || !
```

A.4.11.10 plt/test/scanner/pass/_conditionals.out

```
EQ
NEQ
LT
GT
LEQ
GEQ
AND
OR
NOT
```

A.4.11.11 plt/test/scanner/pass/_control_flow.test

```
if else while for return main
```

A.4.11.12 plt/test/scanner/pass/_control_flow.out

```
IF
ELSE
WHILE
FOR
RETURN
ID
```

A.4.11.13 plt/test/scanner/pass/_delimiters.test

```
( ) { } [ ]
```

A.4.11.14 plt/test/scanner/pass/_delimiters.out

```
LPAREN
RPAREN
LBRACE
RBRACE
LBRACKET
RBRACKET
```

A.4.11.15 plt/test/scanner/pass/_function.test

```
float function (int a, int b) {return 1.0;}
```

A.4.11.16 plt/test/scanner/pass/_function.out

```
FLOAT
ID
LPAREN
```

```
INT
ID
COMMA
INT
ID
RPAREN
LBRACE
RETURN
NUM_LIT
SEMI
RBRACE
```

A.4.11.17 plt/test/scanner/pass/_identifier.test

```
Hey hello number hi123 hello_hi
```

A.4.11.18 plt/test/scanner/pass/_identifier.out

```
ID
ID
ID
ID
ID
```

A.4.11.19 plt/test/scanner/pass/_literal.test

```
"string lit"
12.12
true
false
null
"\"quotes string\""
4
```

A.4.11.20 plt/test/scanner/pass/_literal.out

```
STRING_LIT
NUM_LIT
TRUE
FALSE
NULL
STRING_LIT
NUM_LIT
```

A.4.11.21 plt/test/scanner/pass/_main_function.test

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

A.4.11.22 plt/test/scanner/pass/_main_function.out

```
INT
ID
LPAREN
RPAREN
LBRACE
```

```
RETURN
NUM_LIT
SEMI
RBRACE
```

A.4.11.23 plt/test/scanner/pass/_matrix.test

```
matrix int a
```

A.4.11.24 plt/test/scanner/pass/_matrix.out

```
MATRIX
INT
ID
```

A.4.11.25 plt/test/scanner/pass/_misc.test

```
; , :
```

A.4.11.26 plt/test/scanner/pass/_misc.out

```
SEMI
COMMA
COLON
```

A.4.11.27 plt/test/scanner/pass/_mixed_arithmetic.test

```
100 - 50.12 * 0.4 / 5 - 6.0
```

A.4.11.28 plt/test/scanner/pass/_mixed_arithmetic.out

```
NUM_LIT
MINUS
NUM_LIT
TIMES
NUM_LIT
DIVIDE
NUM_LIT
MINUS
NUM_LIT
```

A.4.11.29 plt/test/scanner/pass/_types.test

```
int float bool void null String true false matrix vector
```

A.4.11.30 plt/test/scanner/pass/_types.out

```
INT
FLOAT
BOOL
VOID
NULL
STRING
TRUE
FALSE
MATRIX
```

VECOR

A.4.12 plt/test/scanner/scripts/build.sh

```
#!/bin/bash

cp ../../src/scanner.mll ./scanner.mll
cp ../../src/parser.mly ./parser.mly
cp ../../src/ast.ml ./ast.ml

ocamllex scanner.mll
ocamlyacc parser.mly
ocamlc -c ast.ml
ocamlc -c parser.mli
ocamlc -c scanner.ml
ocamlc -c parser.ml
ocamlc -c tokenize.ml
ocamlc -o tokenize parser.cmo scanner.cmo tokenize.cmo
```

A.4.13 plt/test/scanner/scripts/clean.sh

```
#!/bin/bash

rm -f *.cmo *.cmi pass/*.res fail/*.res parser.* scanner.* ast.* tokenize
```

A.4.14 plt/test/scanner/scripts/test.sh

```
cat pass/_base_scanner.test | ./tokenize > pass/_base_scanner.res
diff pass/_base_scanner.out pass/_base_scanner.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|      SCANNER: FIRST TEST PASSED       |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|      SCANNER: FIRST TEST FAILED       |"
  echo "----------------------------------------"
fi

cat pass/_delimiters.test | ./tokenize > pass/_delimiters.res
diff pass/_delimiters.out pass/_delimiters.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|   SCANNER: DELIMITERS TEST PASSED     |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|   SCANNER: DELIMITERS TEST FAILED     |"
  echo "----------------------------------------"
```

```
fi

cat pass/_control_flow.test | ./tokenize > pass/_control_flow.res
diff pass/_control_flow.out pass/_control_flow.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|  SCANNER: CONTROL FLOW TEST PASSED    |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|  SCANNER: CONTROL FLOW TEST FAILED    |"
  echo "----------------------------------------"
fi

cat pass/_conditionals.test | ./tokenize > pass/_conditionals.res
diff pass/_conditionals.out pass/_conditionals.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|  SCANNER: CONDITIONALS TEST PASSED    |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|  SCANNER: CONDITIONALS TEST FAILED    |"
  echo "----------------------------------------"
fi

cat pass/_arithmetic.test | ./tokenize > pass/_arithmetic.res
diff pass/_arithmetic.out pass/_arithmetic.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|   SCANNER: ARITHMETIC TEST PASSED     |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|   SCANNER: ARITHMETIC TEST FAILED     |"
  echo "----------------------------------------"
fi

cat pass/_types.test | ./tokenize > pass/_types.res
diff pass/_types.out pass/_types.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|      SCANNER: TYPES TEST PASSED       |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
```

```
  echo "----------------------------------------"
  echo "|      SCANNER: TYPES TEST FAILED       |"
  echo "----------------------------------------"
fi

cat pass/_matrix.test | ./tokenize > pass/_matrix.res
diff pass/_matrix.out pass/_matrix.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|     SCANNER: MATRIX TEST PASSED       |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|     SCANNER: MATRIX TEST FAILED       |"
  echo "----------------------------------------"
fi

cat pass/_comment.test | ./tokenize > pass/_comment.res
diff pass/_comment.out pass/_comment.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|     SCANNER: COMMENTS TEST PASSED     |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|     SCANNER: COMMENTS TEST FAILED     |"
  echo "----------------------------------------"
fi

cat pass/_identifier.test | ./tokenize > pass/_identifier.res
diff pass/_identifier.out pass/_identifier.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|    SCANNER: IDENTIFIER TEST PASSED    |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|    SCANNER: IDENTIFIER TEST FAILED    |"
  echo "----------------------------------------"
fi

cat pass/_mixed_arithmetic.test | ./tokenize > pass/_mixed_arithmetic.res
diff pass/_mixed_arithmetic.out pass/_mixed_arithmetic.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "| SCANNER: MIXED ARITHMETIC TEST PASSED |"
```

```
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "| SCANNER: MIXED ARITHMETIC TEST FAILED  |"
  echo "----------------------------------------"
fi

cat pass/_literal.test | ./tokenize > pass/_literal.res
diff pass/_literal.out pass/_literal.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|     SCANNER: LITERAL TEST PASSED      |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|     SCANNER: LITERAL TEST FAILED      |"
  echo "----------------------------------------"
fi

cat pass/_assignment.test | ./tokenize > pass/_assignment.res
diff pass/_assignment.out pass/_assignment.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|    SCANNER: ASSIGNMENT TEST PASSED    |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|    SCANNER: ASSIGNMENT TEST FAILED    |"
  echo "----------------------------------------"
fi

cat pass/_main_function.test | ./tokenize > pass/_main_function.res
diff pass/_main_function.out pass/_main_function.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|   SCANNER: MAIN FUNCTION TEST PASSED  |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|   SCANNER: MAIN FUNCTION TEST FAILED  |"
  echo "----------------------------------------"
fi

cat pass/_function.test | ./tokenize > pass/_function.res
diff pass/_function.out pass/_function.res > /dev/null
if [ $? = 0 ]; then
```

```
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|    SCANNER: FUNCTION TEST PASSED      |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|    SCANNER: FUNCTION TEST FAILED      |"
  echo "----------------------------------------"
fi

cat pass/_misc.test | ./tokenize > pass/_misc.res
diff pass/_misc.out pass/_misc.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|  SCANNER: MISCELLANEOUS TEST PASSED   |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|  SCANNER: MISCELLANEOUS TEST FAILED   |"
  echo "----------------------------------------"
fi

cat fail/_illegal_carrot.test | ./tokenize >& fail/_illegal_carrot.res
diff fail/_illegal_carrot.out fail/_illegal_carrot.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|      SCANNER: ^ FAIL TEST PASSED      |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|      SCANNER: ^ FAIL TEST FAILED      |"
  echo "----------------------------------------"
fi

cat fail/_illegal_dollar.test | ./tokenize >& fail/_illegal_dollar.res
diff fail/_illegal_dollar.out fail/_illegal_dollar.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|      SCANNER: $ FAIL TEST PASSED      |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|      SCANNER: $ FAIL TEST FAILED      |"
  echo "----------------------------------------"
fi
```

```
cat fail/_illegal_percent.test | ./tokenize >& fail/_illegal_percent.res
diff fail/_illegal_percent.out fail/_illegal_percent.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|      SCANNER: % FAIL TEST PASSED      |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|      SCANNER: % FAIL TEST FAILED      |"
  echo "----------------------------------------"
fi

cat fail/_illegal_period.test | ./tokenize >& fail/_illegal_period.res
diff fail/_illegal_period.out fail/_illegal_period.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|      SCANNER: . FAIL TEST PASSED      |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|      SCANNER: . FAIL TEST FAILED      |"
  echo "----------------------------------------"
fi

cat fail/_illegal_pound.test | ./tokenize >& fail/_illegal_pound.res
diff fail/_illegal_pound.out fail/_illegal_pound.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|      SCANNER: # FAIL TEST PASSED      |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|      SCANNER: # FAIL TEST FAILED      |"
  echo "----------------------------------------"
fi

cat fail/_illegal_tilde.test | ./tokenize >& fail/_illegal_tilde.res
diff fail/_illegal_tilde.out fail/_illegal_tilde.res > /dev/null
if [ $? = 0 ]; then
  echo -e "\e[0;32m"
  echo "----------------------------------------"
  echo "|      SCANNER: ~ FAIL TEST PASSED      |"
  echo "----------------------------------------"
else
  echo -e "\e[0;31m"
  echo "----------------------------------------"
  echo "|      SCANNER: ~ FAIL TEST FAILED      |"
  echo "----------------------------------------"
```

```
fi
```

A.4.15 plt/test/scanner/tokenize.ml

```
open Parser

type num =
  | Int_lit of int
  | Float_lit of float

let stringify = function
  (* Punctuation *)
  | LPAREN -> "LPAREN"  | RPAREN -> "RPAREN"
  | LBRACE -> "LBRACE"  | RBRACE -> "RBRACE"
  | COMMA -> "COMMA"

  (* Arithmetic Operators *)
  | PLUS -> "PLUS"     | MINUS -> "MINUS"
  | TIMES -> "TIMES"   | DIVIDE -> "DIVIDE"

  (* Relational Operators *)
  | EQ -> "EQ"    | NEQ -> "NEQ"
  | LEQ -> "LEQ"  | GEQ -> "GEQ"

  (* Logical Operators & Keywords *)
  | AND -> "AND"   | OR -> "OR"
  | NOT -> "NOT"

  (* Assignment Operator *)
  | ASSIGN -> "ASSIGN"

  (* Conditional Operators *)
  | IF -> "IF"
  | ELSE -> "ELSE"
  (*| ELSEIF -> "ELSEIF"*)

  (* End-of-File *)
  | EOF -> "EOF"

  (* Identifiers *)
  | ID(string) -> "ID"
  | ROWS -> "ROWS" | COLS -> "COLS" | LEN -> "LEN" | TRANSPOSE -> "TRANSPOSE"
  | BAR -> "BAR" | NEW -> "NEW" | FREE -> "FREE"

  (* Literals *)
  | NUM_LIT(num) -> "NUM_LIT"
  | STRING_LIT(string) -> "STRING_LIT"
  | SEMI -> "SEMI" | LBRACKET -> "LBRACKET" | RBRACKET -> "RBRACKET"
  | LT -> "LT" | GT -> "GT" | INC -> "INC" | DEC -> "DEC"
  | COLON -> "COLON" | FOR -> "FOR" | WHILE -> "WHILE"
  | RETURN -> "RETURN"
  | TRUE -> "TRUE" | FALSE -> "FALSE"
```

```
  | INT -> "INT" | BOOL -> "BOOL" | VOID -> "VOID" | FLOAT -> "FLOAT"
  | STRING -> "STRING" |NULL -> "NULL" | MATRIX -> "MATRIX" | VECTOR -> "VECTOR"

let _ =
  let lexbuf = Lexing.from_channel stdin in
  let rec print_tokens = function
    | EOF -> " "
    | token ->
      print_endline (stringify token);
      print_tokens (Scanner.token lexbuf) in
  print_tokens (Scanner.token lexbuf)
```

A.5 plt/.travis-ci.sh

```
export PATH="/usr/bin:$PATH"

sudo apt-get install -qq m4 llvm software-properties-common
sudo add-apt-repository --yes ppa:avsm/ppa
sudo apt-get update -qq
sudo apt-get install -qq -y ocaml ocaml-native-compilers menhir opam
opam init -a
eval `opam config env`
opam install -y depext
opam depext llvm.3.4
opam install -y llvm.3.4 ocamlfind ocamlbuild

cd ./test/scanner
./scripts/build.sh
./scripts/test.sh
./scripts/clean.sh
cd ../parser
./scripts/test.sh
./scripts/clean.sh
cd ../compiler
./scripts/build.sh > build.log
./scripts/test.sh
./scripts/clean.sh
cd ../../hello_world
./scripts/build.sh > build.log
./cmat.native -c hello_world.cmat hello_world.ll
lli hello_world.ll > hello_world.res
diff -q hello_world.out hello_world.res
./scripts/clean.sh

exit 0
```

A.6 plt/hello_world_demo.sh

```
#/usr/bin/sh
read -p "Press [Enter] key to start Hello World Demo"
cd hello_world
```

```
echo "Compiling source"
./scripts/build.sh > build.log
echo "Compilation complete"

./cmat.native -c hello_world.cmat hello_world.ll
echo ""
read -p "Press [Enter] key to run Hello World"
echo ""
lli hello_world.ll
echo ""

echo "Cleaning up"
./scripts/clean.sh
echo "Presentation done"
```

A.7 plt/Makefile

```
MAKE = ./scripts/build.sh
CLEAN = ./scripts/clean.sh

default:
		@$(MAKE)

clean:
		@$(CLEAN)
```

A.8 plt/stdlib.cmat

```
/*
    Vector Functions
*/

    int doti2(vector int [2] v1, vector int [2] v2) {
        int i;
        int sum;
        sum = 0;
        for(i=0; i<v1:len;++i) {
            sum = sum + v1[i]*v2[i];
        }
        return sum;
    }

    int doti3(vector int [3] v1, vector int [3] v2) {
        int i;
        int sum;
        sum = 0;
        for(i=0; i<v1:len;++i) {
            sum = sum + v1[i]*v2[i];
        }
        return sum;
    }
```

```
int doti4(vector int [4] v1, vector int [4] v2) {
    int i;
    int sum;
    sum = 0;
    for(i=0; i<v1:len;++i) {
        sum = sum + v1[i]*v2[i];
    }
    return sum;
}

int doti5(vector int [5] v1, vector int [5] v2) {
    int i;
    int sum;
    sum = 0;
    for(i=0; i<v1:len;++i) {
        sum = sum + v1[i]*v2[i];
    }
    return sum;
}

int doti6(vector int [6] v1, vector int [6] v2) {
    int i;
    int sum;
    sum = 0;
    for(i=0; i<v1:len;++i) {
        sum = sum + v1[i]*v2[i];
    }
    return sum;
}

int doti7(vector int [7] v1, vector int [7] v2) {
    int i;
    int sum;
    sum = 0;
    for(i=0; i<v1:len;++i) {
        sum = sum + v1[i]*v2[i];
    }
    return sum;
}

int doti8(vector int [8] v1, vector int [8] v2) {
    int i;
    int sum;
    sum = 0;
    for(i=0; i<v1:len;++i) {
        sum = sum + v1[i]*v2[i];
    }
    return sum;
}

int doti9(vector int [9] v1, vector int [9] v2) {
    int i;
    int sum;
```

```
    sum = 0;
    for(i=0; i<v1:len;++i) {
        sum = sum + v1[i]*v2[i];
    }
    return sum;
}

float dotf2(vector float [2] v1, vector float [2] v2) {
    int i;
    float sum;
    sum = 0.0;
    for(i=0; i<v1:len;++i) {
        sum = sum + v1[i]*v2[i];
    }
    return sum;
}

float dotf3(vector float [3] v1, vector float [3] v2) {
    int i;
    float sum;
    sum = 0.0;
    for(i=0; i<v1:len;++i) {
        sum = sum + v1[i]*v2[i];
    }
    return sum;
}

float dotf4(vector float [4] v1, vector float [4] v2) {
    int i;
    float sum;
    sum = 0.0;
    for(i=0; i<v1:len;++i) {
        sum = sum + v1[i]*v2[i];
    }
    return sum;
}

float dotf5(vector float [5] v1, vector float [5] v2) {
    int i;
    float sum;
    sum = 0.0;
    for(i=0; i<v1:len;++i) {
        sum = sum + v1[i]*v2[i];
    }
    return sum;
}

float dotf6(vector float [6] v1, vector float [6] v2) {
    int i;
    float sum;
    sum = 0.0;
    for(i=0; i<v1:len;++i) {
        sum = sum + v1[i]*v2[i];
    }
```

```
        return sum;
    }

    float dotf7(vector float [7] v1, vector float [7] v2) {
        int i;
        float sum;
        sum = 0.0;
        for(i=0; i<v1:len;++i) {
            sum = sum + v1[i]*v2[i];
        }
        return sum;
    }

    float dotf8(vector float [8] v1, vector float [8] v2) {
        int i;
        float sum;
        sum = 0.0;
        for(i=0; i<v1:len;++i) {
            sum = sum + v1[i]*v2[i];
        }
        return sum;
    }

    float dotf9(vector float [9] v1, vector float [9] v2) {
        int i;
        float sum;
        sum = 0.0;
        for(i=0; i<v1:len;++i) {
            sum = sum + v1[i]*v2[i];
        }
        return sum;
    }

/*
    Math Functions
*/

    int mod(int dividend, int divisor) {
        int m;
        m = dividend - dividend / divisor * divisor;
        return m;
    }

    int powi(int i, int n) {
        int k;
        int pow;
        pow = 1;
        for(k=0; k<n; ++k) {
            pow = pow*i;
        }
        return pow;
    }

    float powf(float f, int n) {
```

```
        int k;
        float pow;
        pow = 1.0;
        for(k=0; k<n; ++k) {
            pow = pow*f;
        }
        return pow;
    }

    /* calculate sin of an angle (x is in degrees) */

    float sin(float x) {
      int n; int i; int y; float z; float sinx; float PI;

      PI = 3.14159;
      x = x * (PI / 180.0);
      z = x;
      sinx = x;
      n = 1;

      for (i=0; i<10; ++i) {
        y = 2 * n * (2 * n + 1);
        z = -z * x * x / y;
        sinx = sinx + z;
        ++n;
      }
      return sinx;
    }

    /* calculate cos of an angle (x is in degrees) */

    float cos(float x) {
      int n; int i; int y; float z; float cosx; float PI;

      PI = 3.14159;
      x = x * (PI / 180.0);
      z = 1.0;
      cosx = z;
      n = 1;

      for (i=0; i<10; ++i) {
        y = 2 * n * (2 * n - 1);
        z = -z * x * x / y;
        cosx = cosx + z;
        ++n;
      }
      return cosx;
    }

/*
    Matrix Manipulation Functions
*/

    /* Return 2 dimensional rotation matrix from an angle theta */
```

```
matrix float[2,2] rotation_2D_mat(float theta) {
  matrix float[2,2] m; float cosx; float sinx;

  cosx = cos(theta);
  sinx = sin(theta);

  m[0,0] = cosx;
  m[0,1] = -sinx;
  m[1,0] = sinx;
  m[1,1] = cosx;

  return m;
}

/*
    Return 3 dimensional rotation matrix from an angle theta and an axes vector.
    The axes vector is of the form | x | y | z | and determines axis of rotation.
    For example, to rotate about the x axis, simply pass in the vector | 1 | 0 | 0 |
*/

matrix float[3,3] rotation_3D_mat(float theta, vector int[3] axes) {
  matrix float[3,3] m; float cosx; float sinx;

  cosx = cos(theta);
  sinx = sin(theta);
  m = [0.0, 0.0, 0.0;
       0.0, 0.0, 0.0;
       0.0, 0.0, 0.0];
  if (axes[0] == 1) {
    m[0,0] = 1.0;
    m[1,1] = cosx;
    m[1,2] = -sinx;
    m[2,1] = sinx;
    m[2,2] = cosx;
  }
  else if (axes[1] == 1) {
    m[0,0] = cosx;
    m[0,2] = -sinx;
    m[2,0] = sinx;
    m[2,2] = cosx;
    m[1,1] = 1.0;
  }
  else if (axes[2] == 1) {
    m[0,0] = cosx;
    m[0,1] = -sinx;
    m[1,0] = sinx;
    m[1,1] = cosx;
    m[2,2] = 1.0;
  }
  else {
    print_string("Must specify axis of rotation.");
  }
```

```
    return m;
}

/*
    Matrix Printing Functions
*/

    void printi2(vector int [2] V) {
        int i;
        for(i=0; i < V:len; ++i) {
            print_int(V[i]);
        }
        print_string("");
    }

    void printi3(vector int [3] V) {
        int i;
        for(i=0; i < V:len; ++i) {
            print_int(V[i]);
        }
        print_string("");
    }

    void printi4(vector int [4] V) {
        int i;
        for(i=0; i < V:len; ++i) {
            print_int(V[i]);
        }
        print_string("");
    }

    void printi5(vector int [5] V) {
        int i;
        for(i=0; i < V:len; ++i) {
            print_int(V[i]);
        }
        print_string("");
    }

    void printi6(vector int [6] V) {
        int i;
        for(i=0; i < V:len; ++i) {
            print_int(V[i]);
        }
        print_string("");
    }

    void printi7(vector int [7] V) {
        int i;
        for(i=0; i < V:len; ++i) {
            print_int(V[i]);
        }
        print_string("");
    }
```

```
void printi8(vector int [8] V) {
    int i;
    for(i=0; i < V:len; ++i) {
        print_int(V[i]);
    }
    print_string("");
}

void printi9(vector int [9] V) {
    int i;
    for(i=0; i < V:len; ++i) {
        print_int(V[i]);
    }
    print_string("");
}

void printf2(vector float [2] V) {
    int i;
    for(i=0; i < V:len; ++i) {
        print_float(V[i]);
    }
    print_string("");
}

void printf3(vector float [3] V) {
    int i;
    for(i=0; i < V:len; ++i) {
        print_float(V[i]);
    }
    print_string("");
}

void printf4(vector float [4] V) {
    int i;
    for(i=0; i < V:len; ++i) {
        print_float(V[i]);
    }
    print_string("");
}

void printf5(vector float [5] V) {
    int i;
    for(i=0; i < V:len; ++i) {
        print_float(V[i]);
    }
    print_string("");
}

void printf6(vector float [6] V) {
    int i;
    for(i=0; i < V:len; ++i) {
        print_float(V[i]);
    }
```

```
    print_string("");
}

void printf7(vector float [7] V) {
    int i;
    for(i=0; i < V:len; ++i) {
        print_float(V[i]);
    }
    print_string("");
}

void printf8(vector float [8] V) {
    int i;
    for(i=0; i < V:len; ++i) {
        print_float(V[i]);
    }
    print_string("");
}

void printf9(vector float [9] V) {
    int i;
    for(i=0; i < V:len; ++i) {
        print_float(V[i]);
    }
    print_string("");
}

void printi22(matrix int [2,2] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi23(matrix int [2,3] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi24(matrix int [2,4] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
```

```
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi25(matrix int [2,5] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi26(matrix int [2,6] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi27(matrix int [2,7] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi28(matrix int [2,8] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi29(matrix int [2,9] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
```

```
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi32(matrix int [3,2] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi33(matrix int [3,3] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi34(matrix int [3,4] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi35(matrix int [3,5] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi36(matrix int [3,6] M) {
    int i;
    int j;
```

```
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi37(matrix int [3,7] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi38(matrix int [3,8] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi39(matrix int [3,9] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi42(matrix int [4,2] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi43(matrix int [4,3] M) {
    int i;
```

```
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi44(matrix int [4,4] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi45(matrix int [4,5] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi46(matrix int [4,6] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi47(matrix int [4,7] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi48(matrix int [4,8] M) {
```

```
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi49(matrix int [4,9] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi52(matrix int [5,2] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi53(matrix int [5,3] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi54(matrix int [5,4] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}
```

```
void printi55(matrix int [5,5] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi56(matrix int [5,6] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi57(matrix int [5,7] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi58(matrix int [5,8] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi59(matrix int [5,9] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}
```

```
void printi62(matrix int [6,2] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi63(matrix int [6,3] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi64(matrix int [6,4] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi65(matrix int [6,5] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi66(matrix int [6,6] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
```

```
}

void printi67(matrix int [6,7] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi68(matrix int [6,8] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi69(matrix int [6,9] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi72(matrix int [7,2] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi73(matrix int [7,3] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
```

```
    }
}

void printi74(matrix int [7,4] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi75(matrix int [7,5] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi76(matrix int [7,6] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi77(matrix int [7,7] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi78(matrix int [7,8] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
```

```
        print_string("");
    }
}

void printi79(matrix int [7,9] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi82(matrix int [8,2] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi83(matrix int [8,3] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi84(matrix int [8,4] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi85(matrix int [8,5] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
```

```
        }
        print_string("");
    }
}

void printi86(matrix int [8,6] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi87(matrix int [8,7] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi88(matrix int [8,8] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi89(matrix int [8,9] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi92(matrix int [9,2] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
```

```
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi93(matrix int [9,3] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi94(matrix int [9,4] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi95(matrix int [9,5] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi96(matrix int [9,6] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi97(matrix int [9,7] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
```

```
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi98(matrix int [9,8] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printi99(matrix int [9,9] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_int(M[i,j]);
        }
        print_string("");
    }
}

void printf22(matrix float [2,2] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf23(matrix float [2,3] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf24(matrix float [2,4] M) {
    int i;
    int j;
```

```
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf25(matrix float [2,5] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf26(matrix float [2,6] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf27(matrix float [2,7] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf28(matrix float [2,8] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf29(matrix float [2,9] M) {
    int i;
```

```
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf32(matrix float [3,2] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf33(matrix float [3,3] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf34(matrix float [3,4] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf35(matrix float [3,5] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf36(matrix float [3,6] M) {
```

```
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf37(matrix float [3,7] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf38(matrix float [3,8] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf39(matrix float [3,9] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf42(matrix float [4,2] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}
```

```
void printf43(matrix float [4,3] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf44(matrix float [4,4] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf45(matrix float [4,5] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf46(matrix float [4,6] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf47(matrix float [4,7] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}
```

```
void printf48(matrix float [4,8] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf49(matrix float [4,9] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf52(matrix float [5,2] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf53(matrix float [5,3] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf54(matrix float [5,4] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
```

```
}

void printf55(matrix float [5,5] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf56(matrix float [5,6] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf57(matrix float [5,7] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf58(matrix float [5,8] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf59(matrix float [5,9] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
```

```
    }
}

void printf62(matrix float [6,2] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf63(matrix float [6,3] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf64(matrix float [6,4] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf65(matrix float [6,5] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf66(matrix float [6,6] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
```

```
        print_string("");
    }
}

void printf67(matrix float [6,7] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf68(matrix float [6,8] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf69(matrix float [6,9] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf72(matrix float [7,2] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf73(matrix float [7,3] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
```

```
        }
        print_string("");
    }
}

void printf74(matrix float [7,4] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf75(matrix float [7,5] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf76(matrix float [7,6] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf77(matrix float [7,7] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf78(matrix float [7,8] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
```

```
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf79(matrix float [7,9] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf82(matrix float [8,2] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf83(matrix float [8,3] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf84(matrix float [8,4] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf85(matrix float [8,5] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
```

```
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf86(matrix float [8,6] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf87(matrix float [8,7] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf88(matrix float [8,8] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf89(matrix float [8,9] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf92(matrix float [9,2] M) {
    int i;
    int j;
```

```
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf93(matrix float [9,3] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf94(matrix float [9,4] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf95(matrix float [9,5] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf96(matrix float [9,6] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf97(matrix float [9,7] M) {
    int i;
```

```
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf98(matrix float [9,8] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}

void printf99(matrix float [9,9] M) {
    int i;
    int j;
    for(i=0; i < M:rows; ++i) {
        for(j=0; j < M:cols; ++j) {
            print_float(M[i,j]);
        }
        print_string("");
    }
}
```