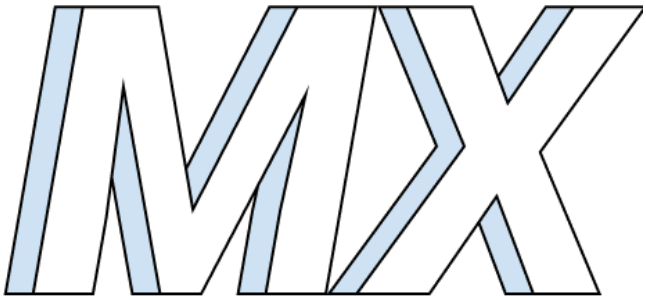


MX

An intuitive and non-imposing Matrix Language

Final Report

Rashel Rojas

Wilderness Oberman

Mauricio Guerrero

Aaron Jackson

December 21, 2021

1. Introduction

Our proposed language, MX, aims to offer programmers an intuitive and efficient means of creating and manipulating matrices.

Although matrices are robust and powerful mathematical structures that are paramount to various fields of Computer Science - attempting to navigate them often results in unnecessary complexities. Moreover, most typical programming languages lack the coherent means of handling matrices without the additional importation of an outside library of some sort. Thus, MX seeks to make matrix processing all the more simpler through providing a streamlined experience of maneuvering matrices.

MX seeks to overhaul the current matrix handling experience by providing one that should be both intuitive and familiar to programmers. MX aims to be intuitive to programmers through its inclusion of the matrix as a data type. By doing this, it hopes to offer users an uncomplicated means of handling matrices that is not too dissimilar from how they might operate more common data types. Moreover, as much of MX follows typical C and Java syntax, it hopes to provide programmers a familiar coding experience that is effortless to pick up on. Programmers will be free to decide for themselves how involved or peripheral they would like MX's matrix handling capabilities to be in their work. Lastly, MX will contain a vigorous built-in library of functions which aims to efficiently automate even the most complex matrix operations. Through implementing standard matrix operations by means of its inclusion as a data type, and providing more intricate manipulations as built-in functions, MX will supply programmers with the components necessary to construct their own complex matrix related functions.

2. Language Tutorial

2.1. Dependencies

Before jumping in, we recommend running this language and all of its associated tests with the following requirements installed:

- Clang / LLVM version 13.0.0
- Ocaml 4.12.0

We don't anticipate any major differences in the functionality of the language but we have noticed that members with different versions of LLVM and OCaml would receive different error messages for Fatal Exceptions - thus causing tests to fail on one machine and pass on another. The above configurations were used to write the tests and all tests should pass with these settings.

2.2. Compiling the MX compiler

Upon unzipping the mx.zip file, cd into the MX/ directory and run:

```
$ make
```

This command should both build the mx.native compiler and also run all of our tests via the .testall.sh script.

2.3. Compiling an MX program

We've provided a script that both builds the mx.native compiler and uses it to compile an argument provided on the command line. To test this script, make a file in your root directory called hello.mx fill it with the following:

```
int main(){
     prints("Hello World!");
     return 0;
}
```

After saving the file, in the root directory, run:

```
./cmx.sh hello.mx
```

This should both build an executable file and also execute it.

2.4. Sample Program

This sample program is provided as a brief tour of the MX syntax. In this snippet of code, you will be able to view examples of Function Declarations, Control Flow, 5 of the 6 MX data types (void excluded), Variable Declarations, Variable Initialization, the MX Matrix Literal, Function calls and Single and Multi line comments. Much of the syntax is based on Micro C.

```
int gcd(int a, int b) {
    while (a != b) {
```

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

int main()
{
    String s; Matrix m; bool b; float f; int a;

    s = "Hello World";
    f = 2.1;
    a = 2;
    b = false;
    m = [[1,2],[3,4]];

    print(gcd(2,14));
    print(gcd(3,15));
    print(gcd(99,121));

    #A single line comment

    /* A  multi line
    comment */
    return 0;
}
```

3. Language Manual

3.1. Lexical conventions

3.1.1. Comments

- The `#` character begins single-line comments
- `/*` begins a multi-line comment and `*/` terminates multi-line comments.

3.1.2. Identifiers

- Identifiers are sequences of letters and digits, in which the first character has to be a letter. Inclusive of lowercase and uppercase letters. Identifiers may contain underscores. The type of the

identifier as well as its initial value must be specified upon declaration of the identifier.

3.1.3. Key words

- The following identifiers are reserved and may not be used otherwise
  - `int`
  - `float`
  - `for`
  - `return`
  - `if`
  - `else`
  - `void`
  - `bool`
  - `String`
  - `False`
  - `Matrix`
  - `while`
  - `True`
- The following identifiers are reserved for function names. A function may not be declared using the following keywords
  - `main`
  - `numRows`
  - `print_matrix`
  - `identity`
  - `printf`
  - `pi`
  - `numCols`
  - `transformation`
  - `prints`
  - `print_matrix`
  - `Printb`
  - `print`

3.1.4. Strings

- A sequence of zero or more characters (lowercase/upper case), digits, enclosed in double quotes. Strings with only one character are still considered to be variables of type String and not type char.

3.2. Primitive Types

3.2.1. Types

- `int` - signed integer type, 4 bytes
- `float` - signed floating-point type, 8 bytes
- `bool` - has the value of true/false, 1 byte
- `void` - used for functions that do not return a value.
- `Matrix` - MX's Matrix data type which consists of elements of type int only.

3.2.2. Implicit Casting on Arithmetic Operations

- If there is an arithmetic operation between an *int* and *float* (i.e. addition, subtraction, multiplication, division), then cast the *int* value to a *float*.

- Note: Assignment (=, +=, -=, *=) and relational (<, >, !=, ==, etc.) operators do not support *int* to *float* casting.

3.3. Operators

3.3.1. Unary Operators

- *`!expr`*
- *`-expr`*

3.3.2. Assignment operator

- Values may be assigned to variables via the following syntax:
  - *`Identifier = expr;`*
  - Where the value in expression will be assigned to the identifier

- The values of the results of arithmetic operations may also be assigned via the following syntax:
  - *`ID += Expr`*
    - Assignment by sum
  - *`ID -= Expr`*
    - Assignment by difference
  - *`ID *= Expr`*
    - Assignment by product
  - Note: Assignment operators (=, +=, -=, *=) do not support *int* to *float* casting

3.3.3. Arithmetic Operators

- *Expr + Expr*
  - The result is the sum of the expressions. This operation may only be performed between expressions of type *int* and type *float*. If performed between an expression of type *int* and another expression of type *float* (i.e. *int* a + *float* b), the *int* is converted to a *float* and the type of the sum is of type *float*.
- *Expr - Expr*
  - The result is the difference of the expressions. The same type considerations for addition apply.
- *Expr * Expr*
  - The result is the product of the expressions. The same type considerations for addition apply.
- *Expr / Expr*
  - The result is the quotient of the expressions. The same type considerations for addition apply.

3.3.4. Matrix Operators

The following section details operators that are specific to MX's *Matrix* data type and the operations that can be performed between multiple expressions of type *Matrix* and, to a lesser extent, type *int.*

- *Matrix +. Matrix*
  - The result is the sum of two expressions of type *Matrix*. This operation may only be performed between two expressions of type *Matrix* whose elements are of type *int*.
- *Matrix -. Matrix*
  - The result is the difference between two expressions of type *Matrix*. This operation may only be performed between two expressions of type *Matrix* whose elements are of type *int*.
- *Matrix *. Matrix*
  - The result is the product of two expressions of type *Matrix*. This operation may only be performed between two expressions of type *Matrix* whose elements are of type *int*.
- *Matrix **. Expr*

    - This operator indicates scalar multiplication between an expression of type *Matrix* and another expression of type *int*. Associativity is irrelevant.
- *Matrix'*
    - This operator returns the transpose of a matrix. Return type is *Matrix*.

3.3.5. Relational Operators

The following operators are reserved for comparison between two expressions. Each operator yields 1 if the comparison is True and 0 if it is false. Comparison between expressions requires that each expression be of the same type. The last two operators have lower precedence than the first four.

- *Expr < Expr*
- *Expr > Expr*
- *Expr <= Expr*
- *Expr >= Expr*
- *Expr == Expr*
- *Expr != Expr*
- Note: Relational operators do not support *int* to *float* casting.

3.3.6. Logic Operators

- *Expr && Expr*
- *Expr || Expr*
- *!Expr*

3.3.7. Order of Precedence

Precedence	Description	Associativity
() []	Grouping or Function call Array subscripting	left-to-right
- ! ‘	Unary minus Logical NOT Transpose	right-to-left
* /	Arithmetic multiplication and division	left-to-right

*. **. %	Matrix-matrix multiplication, scalar-matrix multiplication Modulus	
+ - +. -.	Arithmetic addition and subtraction Matrix addition and subtraction	left-to-right
< <= > >=	Less than, less than or equal to Greater than, greater than or equal to	left-to-right
== !=	Is equal to, is not equal to	left-to-right
&&	Logical AND	left-to-right
||	Logical OR	left-to-right
= += -= *=	Assignment Assignment operators	right-to-left
,	Separates expressions	left-to-right

3.4. Separators

3.4.1. Semicolon at the end of every statement (not including end of for/while loop blocks, if/else statements)

3.4.2. Curly braces in for loops, while loops, if//else

3.4.3. ( ) [ ] { } ; , .

3.4.4. Ignore whitespace (don't make it a token)

3.5. Declarations

All variables should be declared with their type specification and identifier only. Users will not be allowed to declare and initialize variables in the same line.

3.5.1. Declaring primitive types:

- `type var_name;`

3.5.2. Initializing variables:

- Variables may only be initialized after they have been declared and not on the same line. For example, an `int` variable `a` that stores a value of 2 may be declared and initialized as such:

- `Int a;`
- `a = 2;`

- The following however is an incorrect method of initializing a variable and will throw an error:
  - `Int a = 2;`

3.5.3. Functions

- All functions must be defined when being declared. Function names include letters/digits (lowercase/uppercase) and when declaring functions, it should specify the return type as shown below in *datatype*:

```
datatype foo(datatype parameter1, … , datatype parameter_n) {

}
```

All functions are public.

- Every MX program should contain a `main()` function, which starts every program.

3.6. Statements

3.6.1. Expression statements

- Expression statements take the form of "*expr;*"

3.6.2. Conditional statements

- Conditional statements may take the forms of:
  - `If ( expr ) { stmt; }`
  - `If ( expr ) { stmt; } else { stmt; }`

3.6.3. While statement

- ```
  while ( expr ) {
  stmt;
  }
  ```

3.6.4. For statement

- ```
  for( expr_1, expr_2, expr_3 ) {
  stmt;
  }
  ```

3.6.5. Return statement

- If a function is of type void (As declared in its declaration), a return statement is not required. Otherwise, a return statement is required.

  `return` *`expr`*`;`

3.7. Scope

Declarations made within functions are visible only within those functions (i.e. their scope). A declaration is not visible to declarations that came before it. You cannot declare an already declared variable, but redefining variables is allowed.

3.8. Function calls

3.8.1. Functions may be called using the following syntax:

- `function_name(parameter_1, … , parameter_n )`

3.9. Sample Code

Basic syntax: example of a user defined function for determining the greatest common divisor of two integers

```
int gcd(int x, int y)
{
     # example of a simple user-defined function
     while (x != y)
     {
          if (x > y)
               x -= y;
          else
               y -= x;
     }
     return x;
}
```

```
int main ()
{
    int x = 3;
    int y = 15;
    int z = gcd(x, y);
    printf("%d", z); # prints 3
    return 0;
}

Simple program illustrating built in declaration and manipulation of
matrices in our language

int main()
{
Matrix m1;
Matrix m2;
Matrix m3;
Matrix m4;

m1 = [[0, 1], [2, 3]]; # matrix declaration
print_matrix(m1);
/* prints the following
[0, 1]
[2, 3]
*/

m2 = [[3, 4], [4, 5]];
      print_matrix(m2);

/* prints the following
[3, 4]
[4, 5]
*/


m3 = m1 *. m2;
print_matrix(m3);
/* prints the following
[4, 5]
```

```
[1, 2]
[8, 3]
*/


m4 = m1' +. m2;
print_matrix(m4);

/* prints the following
[3, 6]
[5, 8]
*/

     return 0;
}
```

C-program approximation of matrix manipulation. As you can see, our language will improve the way in which matrices are added, subtracted, etc. (less lines of code).

```
#include <stdio.h>
#include <stdlib.h>

void add(int m[2][2], int n[2][2], int sum[2][2])
{
    for(int i = 0; i < 2; i++)
        for(int j = 0; j < 2; j++)
            sum[i][j] = m[i][j] + n[i][j];
}

void multiply(int m[2][2], int n[2][2], int res[2][2])
{
    for(int i = 0; i < 2; i++)
    {
        for(int j = 0; j < 2; j++)
        {
            res[i][j] = 0;
            for (int k = 0; k < 2; k++)
                res[i][j] += m[i][k] * n[k][j];
```

```
        }
    }
}

void transpose(int matrix[2][2], int trans[2][2])
{
    for (int i = 0; i < 2; i++)
        for (int j = 0; j < 2; j++)
            trans[i][j] = matrix[j][i];
}

void print_matrix(int matrix[2][2])
{
    for(int i = 0; i < 2; i++)
    {
        printf("[");
        for(int j = 0; j < 2; j++)
        {
            printf("%d", matrix[i][j]);
            if(j < 1)
                printf("\t");
        }
        printf("]\n");
    }
}

int main()
{
    int m1[2][2] = {{0, 1},{2, 3}};
    int m2[2][2] = {{3, 4},{4, 5}};
    int m3[2][2];
    print_matrix(m1);
    printf("\n");
    print_matrix(m2);
    printf("\n");
    multiply(m1, m2, m3);
    print_matrix(m3);
    printf("\n");
    transpose(m1, m3);
```

```
    add(m3, m2, m3);
    print_matrix(m3);
    printf("\n");

    return 0;
}
```

4. Project Plan

4.1. Processes

4.1.1. Planning

Our MX Team was formed directly after the first lecture. We had not known each other prior to this course and immediately set up a when2meet to decide on an initial meeting time. After the initial meeting, we set a new time for a weekly meeting, logged it in Google Calendar which we also populated with other important events and deadlines, set up a shared Google Drive folder for any project related documents (such as this Final Report) and began brainstorming ideas for possible languages. As some members of the group had familiarity with matrices and coded implementations of them, and we figured that there would be a plethora of resources at our disposal as many other groups would have made matrix languages in the past, we decided to settle on a Matrix language (MX) for our own project.

Soon, we ended up also setting another scheduled weekly meeting time - which would take place directly after our weekly discussion with our TA and allow us to debrief on any feedback we received. As deadlines approached however, much of our scheduled meeting times were exchanged for more frequent and spontaneous gatherings as we all thought it best to work on the project together. Thus, much of the earlier deliverables that defined what the language would look like such as the proposal, LRM, Parser and AST were worked on together, whereas other implementations that dealt moreso with actually implementing the specifications we defined were worked on individually.

4.1.2. Specification

Our specification process ended up being somewhat mild. At its very core, MX is a C-like language with a Matrix data type and that is the language that, more or less, we were able to make. Specification would come in the

form however of editing our LRM to more closely resemble the syntax and language functionality that we actually ended up / had the time to implement. Getting the Matrix data type to work and mesh well with the language was a priority so other features such as a variety of assignment operators or declaring and initializing a variable in the same line had, sadly, had to be cut.

4.1.3. Development

For early deliverables such as the Scanner, Parser and AST, we all thought that it would be best to code these up together as these interfaces would define the general structure of the MX language. For the Hello World deliverable, we all worked individually to “figure out” the MicroC architecture and how we could adopt it for our own language. Once one of us had figured it out, we all regrouped and worked together on getting our mx.native compiler to actually compile a simple MX source file and ended up being able to compile and execute gcd.mx - which we submitted for our hello world deliverable. Afterwards, we iterated on our Parser and AST multiple times to better tailor them to how we’d like our language to be handled in the back-end files (Semant.ml, Codegen, etc). Once done, we worked individually on reconfiguring these same back-end files to accommodate the Matrix datatype and its functionality. Once we had a basic pipeline up and running, we were able to begin writing our Matrix library and simultaneously write tests for it as we went along.

4.1.4. Testing

Because our language is very similar to the C language, we began our project by borrowing lines from the MicroC project. When developing the Scanner and Parser for our language, we used Ocamllex and Ocamlyacc, respectively, to confirm that they were working correctly. When we began adding more features, we tested our code by running simple examples such as printing strings, printing matrices, etc. Once we started writing built-in functions for matrices, we tested each one with test files that were run as a collection via a test script, testall.sh. We created passing and failing test cases and had the script compare their outputs to the expected outputs in the .out and .err files in our test suite. We also added test files for other components of our language such as if blocks, variable assignment, user-created functions, etc.

4.2. Style Guide

As all files were either worked on with everyone present or with only 1-2 people assigned to a file, we did not determine it necessary to have a style guide. Moreover, and a bit more honestly, we also did not have the foresight to develop one.

We however, did have some general rules that were informally acknowledged and agreed upon. These rules are as follows:

Ocaml

- Single line “let-in” statements may go on the same line
- For Multi-line “let-in” statements, “let” and “in” should occupy their own lines and all content between them must be indented by one tab.
- All arrows for a pattern matching should be vertically aligned.
- If the content following an arrow in a pattern match is multi-lined, then the arrow should be on its own line and the proceeding content should be indented to be vertically aligned with the beginning of the content of all other pattern matches.
- *Attempt* to align all vertical lines of each pattern match case.
- Be mindful to comment seemingly trivial code.

C

- Follow Standard indentation guidelines regarding braces.
- Matrix function names should be analogous to their names in codegen.ml
- Variable and function names should be camel case.

General

- Make meaningful commit messages.
- Alert everyone once you’ve pushed to the Repo.
- Functionality over style - get the code working whatever way makes sense to you then format later if possible.
- Have fun!

4.3. Timeline

Oct. 3rd - Oct. 9th	Language Proposal developed and submitted.
Oct. 24th - Oct. 31st	Repository made; LRM developed and submitted; began working on parser.
Oct. 31st - Nov. 6th	Parser complete; Scanner complete.
Nov. 7th - Nov. 13th	Completed hello world deliverable; developed bash script to run hello world deliverable;
Nov. 14th - Nov. 20th	Included Print String support;
Nov. 21st - Nov. 27th	Iterated on Parser Matrix declaration rules;
Nov. 28th - Dec. 4th	Linked Matrix Library; General pipeline for Matrix datatype up and running; Makefile updated to incorporate C library.
Dec. 5th - Dec. 11th	Can now initialize Matrices; Semant.ml error checks on matrix literals; SMx structure changed; More Matrix functions added to the Matrix library.
Dec. 12th - Dec. 19th	More functions added to mx.c; Matrix operators implemented; testing suite in development; Final proposal, slides and demo program in development
Dec. 20th - Dec. 22nd	Fully developed Language, Final Report and Presentation.

4.4. Roles and Responsibility

For better or worse, team MX never decided on clearly defined roles and, instead, decided on a structure wherein we would all help out where we could. For a

somewhat clearer image of what team members actually ended up working on, please reference the following table:

Member	**Responsibilities**
Aaron Jackson	Language Guru, Co-manager, System Architect
Rashel Rojas	Language Guru, Co-manager, Tester
Wilderness Oberman	Language Guru, Matrix Expert
Mauricio Guerrero	Language Guru, Tester

4.5. Tools

4.5.1. **Languages**: Ocaml 4.12.0, LLVM / Clang 13.0.0., C, bash

4.5.2. **Environment**: Linux

4.5.3. **Version Contro**l: Github

4.5.4. **Code Editor**: Visual Studio Code

4.5.5. **Documentation**: Google Drive Suite

4.5.6. **Operating Systems**: Windows

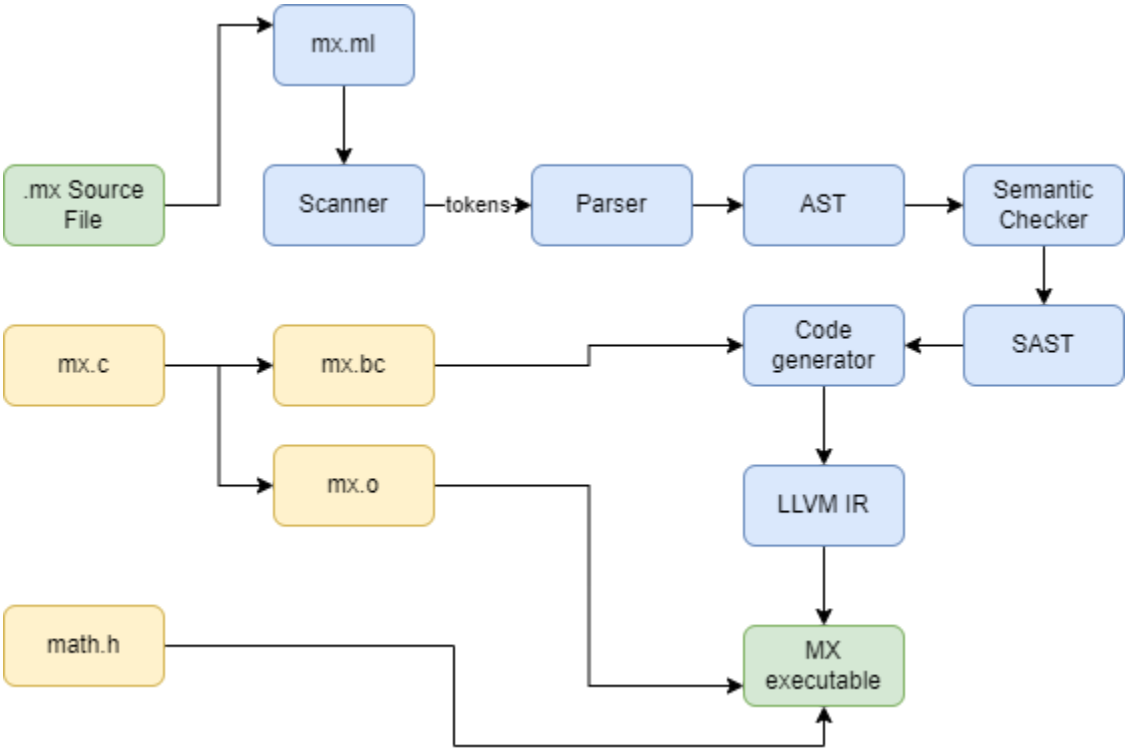
4.5.7. **Communication**: SMS

4.6. Project Log

Please see appendix for project log

5. Architectural Design

5.1. Architecture Overview



5.2. mx.ml (Aaron)

mx.ml compiles into the mx.native compiler / driver file that calls the scanner, parser, semant and codegen to act on the .mx source file and translate it into LLVM IR.

5.3. Scanner.ml (Rashel, Mauricio, Wilderness, Aaron)

The scanner is the first interface that acts upon the .mx source code. The scanner "scans" over the code and performs lexical analysis through aggregating the inputted code into tokens which are then passed on to MX's parser. If the scanner passes over any illegal characters it will reject it and throw an error.

5.4. Parser.mly (Rashel, Mauricio, Wilderness, Aaron)

MX's parser is based upon an unambiguous context-free grammar for the arrangement of tokens found in the source code of MX source files. The parser

receives these tokens from the scanner and, based on their arrangement, uses them to generate an abstract syntax tree. If there are any irregular

5.5. Ast.ml (Rashel, Mauricio, Wilderness, Aaron)

MX's abstract syntax tree represents the structure of the code of a MX source file.

5.6. Semant.ml (Aaron)

MX's Semant file parses through the AST produced by the Parser and performs a semantic analysis on it to ensure that it is semantically correct. While performing semantic analysis, the semantic checker extends and translates the AST into an analogous SAST which features information regarding types. If the AST is semantically correct, the semantic checker then outputs an SAST.

5.7. Sast.ml (Aaron)

MX's semantic abstract syntax tree represents the structure of the code of an MX source file with typing information included.

5.8. Codegen.ml (Aaron)

Once provided the SAST, MX's codegen is responsible for generating the LLVM IR code of an MX source file. The codegen also links in a bitcode (.bc) build of MX's C matrix library (mx.c) to represent the Matrix data type.

5.9. mx.c (Wilderness, Aaron)

mx.c is MX's Matrix library built in C. This library contains information about the Matrix datatype as well as various functions and operations for the Matrix data type. A bitcode build of this file (mx.bc) is linked in the codegen and an output build (mx.o) is linked with the MX executable file once the C compiler has been called on the machine code implementation of the MX source file.

6. Test Plan

6.1. Two Representative Programs

The program below shows a simple example of matrix declaration, the transpose of a matrix, and the printing of matrices. The apostrophe on the sixth line indicates the transpose operation, in which the rows of matrix **m** will become the columns of matrix **n**.

```
int main() {
    Matrix m;
    Matrix n;

    m = [[2,4],[6,8],[10,12]];
    n = m';
    print_matrix(m);
    print_matrix(n);
    return 0;
}
```

Below is the LLVM file generated for this program (sample-program1.ll).

```
; ModuleID = 'MX'
source_filename = "MX"

%struct.Matrix = type { i32, i32, i32*, i32 }

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00", align 1
@fmt.1 = private unnamed_addr constant [4 x i8] c"%s\0A\00", align 1
@fmt.2 = private unnamed_addr constant [4 x i8] c"%g\0A\00", align 1

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

declare i32 @printbig(i32)

declare %struct.Matrix* @initMatrix(i32, i32)

declare %struct.Matrix* @store(%struct.Matrix*, i32)

declare %struct.Matrix* @display(%struct.Matrix*)

declare %struct.Matrix* @transpose(%struct.Matrix*)

declare %struct.Matrix* @mxAdd(%struct.Matrix*, %struct.Matrix*)

declare %struct.Matrix* @mxSub(%struct.Matrix*, %struct.Matrix*)

declare %struct.Matrix* @mxMult(%struct.Matrix*, %struct.Matrix*)

declare %struct.Matrix* @identity(i32)

declare %struct.Matrix* @mxScale(%struct.Matrix*, i32)
```

```
declare %struct.Matrix* @twoFunc(i32, i32)

declare %struct.Matrix* @transformation(%struct.Matrix*, i32)

declare i32 @numCols(%struct.Matrix*)

declare i32 @numRows(%struct.Matrix*)

declare double @pi()

define i32 @main() {
entry:
  %m = alloca %struct.Matrix*, align 8
  %n = alloca %struct.Matrix*, align 8
  %init_matrix = call %struct.Matrix* @initMatrix(i32 3, i32 2)
  %store_matrix = call %struct.Matrix* @store(%struct.Matrix* %init_matrix, i32 2)
  %store_matrix1 = call %struct.Matrix* @store(%struct.Matrix* %init_matrix, i32 4)
  %store_matrix2 = call %struct.Matrix* @store(%struct.Matrix* %init_matrix, i32 6)
  %store_matrix3 = call %struct.Matrix* @store(%struct.Matrix* %init_matrix, i32 8)
  %store_matrix4 = call %struct.Matrix* @store(%struct.Matrix* %init_matrix, i32
10)
  %store_matrix5 = call %struct.Matrix* @store(%struct.Matrix* %init_matrix, i32
12)
  store %struct.Matrix* %init_matrix, %struct.Matrix** %m, align 8
  %m6 = load %struct.Matrix*, %struct.Matrix** %m, align 8
  %transpose = call %struct.Matrix* @transpose(%struct.Matrix* %m6)
  store %struct.Matrix* %transpose, %struct.Matrix** %n, align 8
  %m7 = load %struct.Matrix*, %struct.Matrix** %m, align 8
  %printbig = call %struct.Matrix* @display(%struct.Matrix* %m7)
  %n8 = load %struct.Matrix*, %struct.Matrix** %n, align 8
  %printbig9 = call %struct.Matrix* @display(%struct.Matrix* %n8)
  ret i32 0
}
```

Below is our second program, which demonstrates some of our matrix operations: addition, subtraction, multiplication, and scalar multiplication.

```
int main() {
    Matrix m1;
    Matrix m2;

    m1 = [[1,2],[3,4]];
    m2 = [[2,4],[6,8]];

    print_matrix(m1 +. m2);
    print_matrix(m1 -. m2);
```

```
    print_matrix(m1 *. m2);
    print_matrix(m2 **. 3);
}
```

The LLVM code generated for this program is shown below.

```
; ModuleID = 'MX'
source_filename = "MX"

%struct.Matrix = type { i32, i32, i32*, i32 }

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00", align 1
@fmt.1 = private unnamed_addr constant [4 x i8] c"%s\0A\00", align 1
@fmt.2 = private unnamed_addr constant [4 x i8] c"%g\0A\00", align 1

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

declare i32 @printbig(i32)

declare %struct.Matrix* @initMatrix(i32, i32)

declare %struct.Matrix* @store(%struct.Matrix*, i32)

declare %struct.Matrix* @display(%struct.Matrix*)

declare %struct.Matrix* @transpose(%struct.Matrix*)

declare %struct.Matrix* @mxAdd(%struct.Matrix*, %struct.Matrix*)

declare %struct.Matrix* @mxSub(%struct.Matrix*, %struct.Matrix*)

declare %struct.Matrix* @mxMult(%struct.Matrix*, %struct.Matrix*)

declare %struct.Matrix* @identity(i32)

declare %struct.Matrix* @mxScale(%struct.Matrix*, i32)

declare %struct.Matrix* @twoFunc(i32, i32)

declare %struct.Matrix* @transformation(%struct.Matrix*, i32)

declare i32 @numCols(%struct.Matrix*)
```

```
declare i32 @numRows(%struct.Matrix*)

declare double @pi()

define i32 @main() {
entry:
  %m1 = alloca %struct.Matrix*, align 8
  %m2 = alloca %struct.Matrix*, align 8
  %init_matrix = call %struct.Matrix* @initMatrix(i32 2, i32 2)
  %store_matrix = call %struct.Matrix* @store(%struct.Matrix* %init_matrix,
i32 1)
  %store_matrix1 = call %struct.Matrix* @store(%struct.Matrix*
%init_matrix, i32 2)
  %store_matrix2 = call %struct.Matrix* @store(%struct.Matrix*
%init_matrix, i32 3)
  %store_matrix3 = call %struct.Matrix* @store(%struct.Matrix*
%init_matrix, i32 4)
  store %struct.Matrix* %init_matrix, %struct.Matrix** %m1, align 8
  %init_matrix4 = call %struct.Matrix* @initMatrix(i32 2, i32 2)
  %store_matrix5 = call %struct.Matrix* @store(%struct.Matrix*
%init_matrix4, i32 2)
  %store_matrix6 = call %struct.Matrix* @store(%struct.Matrix*
%init_matrix4, i32 4)
  %store_matrix7 = call %struct.Matrix* @store(%struct.Matrix*
%init_matrix4, i32 6)
  %store_matrix8 = call %struct.Matrix* @store(%struct.Matrix*
%init_matrix4, i32 8)
  store %struct.Matrix* %init_matrix4, %struct.Matrix** %m2, align 8
  %m19 = load %struct.Matrix*, %struct.Matrix** %m1, align 8
  %m210 = load %struct.Matrix*, %struct.Matrix** %m2, align 8
  %mxAdd = call %struct.Matrix* @mxAdd(%struct.Matrix* %m19,
%struct.Matrix* %m210)
  %printbig = call %struct.Matrix* @display(%struct.Matrix* %mxAdd)
  %m111 = load %struct.Matrix*, %struct.Matrix** %m1, align 8
  %m212 = load %struct.Matrix*, %struct.Matrix** %m2, align 8
  %mxSub = call %struct.Matrix* @mxSub(%struct.Matrix* %m111,
%struct.Matrix* %m212)
  %printbig13 = call %struct.Matrix* @display(%struct.Matrix* %mxSub)
  %m114 = load %struct.Matrix*, %struct.Matrix** %m1, align 8
  %m215 = load %struct.Matrix*, %struct.Matrix** %m2, align 8
  %mxMult = call %struct.Matrix* @mxMult(%struct.Matrix* %m114,
%struct.Matrix* %m215)
```

```
  %printbig16 = call %struct.Matrix* @display(%struct.Matrix* %mxMult)
  %m217 = load %struct.Matrix*, %struct.Matrix** %m2, align 8
  %mxScale = call %struct.Matrix* @mxScale(%struct.Matrix* %m217, i32 3)
  %printbig18 = call %struct.Matrix* @display(%struct.Matrix* %mxScale)
  ret i32 0
}
```

6.2. Test Suite and Automation

Note: All of our tests and their respective outputs can be found in the appendix.

Our testing script, `testall.sh`, was adopted from the MicroC compiler and modified for our language. We used it to automatically run all of the tests in our test suite.

Our test suite can be found in our `tests/` folder and consists of three types of files: `.mx`, `.out` , and `.err`. Test cases that we expect to compile and run successfully begin with "`test-`" and those that we do not begin with "`fail-`". Our testing script works by comparing the output of each `.mx` file to the expected output, which can be found in the corresponding `.out` and `.err` files for successful and failing test cases, respectively.

We chose test cases such that they provided a comprehensive view into all of the possible exceptions that a typical coder using our language would encounter and normal cases that that user would create. Our test cases can be broken down into these main categories:

1. Variable declaration
2. Arithmetic operations on integers and floats
3. Matrix declaration and operations
4. Control flow
5. Function creation
6. Logical/relational operators
7. Printing

When our testing script is run, this is the expected result. The "OK" next to each test case indicates that it ran successfully (the output matches what is in the corresponding `.out`/`.err` file).

```
clang -emit-llvm -o mx.bc -c mx.c -Wno-varargs
opam exec -- \
ocamlbuild -use-ocamlfind mx.native -pkgs llvm,llvm.analysis,llvm.bitreader
Finished, 25 targets (0 cached) in 00:00:01.
cc    -c -o mx.o mx.c
./testall.sh
test-arithmetic...OK
test-assignment...OK
test-assignop...OK
test-for...OK
test-func...OK
test-gcd...OK
test-hello...OK
test-if...OK
test-logicalops...OK
test-matrixAdd...OK
test-matrixIdentity...OK
test-matrixMul2...OK
test-matrixMul...OK
test-matrix...OK
test-matrixRotation...OK
test-matrixRowsColumns...OK
test-matrixScalar...OK
test-matrixSub...OK
test-matrixTransformation...OK
test-matrixTranspose...OK
test-relationalops...OK
test-sampleprogram1...OK
test-sampleprogram2...OK
test-transpose1...OK
test-while...OK
fail-add1...OK
fail-add2...OK
fail-add3...OK
fail-add4...OK
fail-arithmetic2...OK
fail-arithmetic3...OK
fail-arithmetic...OK
fail-assign1...OK
fail-assign2...OK
fail-assign3...OK
fail-assignop...OK
fail-dead...OK
fail-for...OK
fail-funcarg1...OK
fail-funcarg2...OK
fail-funcarg3...OK
```

```
fail-identity1...OK
fail-identity2...OK
fail-if...OK
fail-matrix-declaration...OK
fail-matrixfunction1...OK
fail-matrixfunction2...OK
fail-mult1...OK
fail-mult2...OK
fail-mult3...OK
fail-mult4...OK
fail-nomain...OK
fail-numcols...OK
fail-numrows...OK
fail-opand...OK
fail-printb...OK
fail-printmatrix1...OK
fail-print...OK
fail-prints...OK
fail-reservedfuncs...OK
fail-scalemult1...OK
fail-scalemult2...OK
fail-subtract1...OK
fail-subtract2...OK
fail-transform1...OK
fail-transform2...OK
fail-transpose1...OK
fail-transpose2...OK
fail-undecvar...OK
fail-while1...OK
```

6.3. Roles

The testing script, `testall.sh`, was adopted from the MicroC compiler and modified for our language by Mauricio. The fail test cases in the `tests/` folder were written by Rashel and the passing test cases were written by Rashel and Mauricio.

7. Lessons Learned

7.1. Aaron Jackson

Speaking candidly, initially, I was not very enthusiastic about this project and just wanted to get it over with for the sake of completion. I was not very interested in understanding how a language operated / functioned and the steep learning curve that I was anticipating also did not do much to spark my interest. And, while most of those things are still true - I think that, the more work and effort I put in to actually understanding the material, the more I was able to actually enjoy the work that I was doing on the project. Even OCaml became somewhat marginally more enjoyable to use as I found myself having to implement it in various areas of our project such as the codegen file.

My piece of advice for anyone that may be as unenthused about this project as I was is to just rip off the band-aid and dive headfirst into the material instead of avoiding it. It will be difficult and annoying at first - but it does get a lot easier and more enjoyable the more you use it or even attempt to use it. If you're like me, you will fail to implement a lot of the things you initially set out to implement - but failing at that one thing may give you insight into implementing an entirely different thing so nothing is truly ever a waste of time! More concretely however - spend a lot of time looking at the LLVM OCaml bindings (and be sure to ask the TAs / Stephen what those actually are). I realized very late into our project just what the point of the llvm.moe page was and I wish I had realized sooner as it would have made implementing features in our language much simpler!

Moreover, I also highly recommend assigning group members specific responsibilities from the get-go instead of leaving things up in the air until someone decides to work on it. We did not assign specific responsibilities and I feel that that resulted in a dynamic where one person would work on completing a task and build domain experience in that area - but since no one else did, they were unable to assist in future tasks regarding that area, which would lead to an uneven distribution of labor. In addition to specific roles, I highly recommend maybe assigning two people to each role (who can maybe pair-program together) so that there is at least one other person with knowledge regarding a topic that can assist / help pick up the workload.

7.2. Wilderness Oberman

When I learned that a group project was a necessary requirement for this class, I don't know which I dreaded more, the idea of working on a group project or

designing a new language. The best advice I can offer is to choose your teammates wisely; I was fortunate enough to find a group that equally divided responsibilities for building our language, with each member working on a part that played on their strengths. Although our original design idea was a bit ambitious, and we wanted to include every single possible matrix function that existed in linear algebra, it was necessary to set realistic goals and be flexible with the language design for the completion of this project. Using the LRM as an overall roadmap to building our language was helpful, but it's important to realize that by the time you finally get your language to compile and complete basic operations, your language may look entirely different than what you had intended. Designing and building a new language can seem like an impossible undertaking, but using the MicroC code as the base of our language was extremely helpful in understanding the complex OCaml code found in the Semant.ml and Codegen. While there have been many times that I have been frustrated that certain functions did not exist in C/C++, I have a better understanding as to why certain design features were/were not implemented. Probably the most important lesson that I learned was that neither OCaml nor C are particularly "friendly" for coding matrix operations and I'd like to warn future PLT students that if you believe writing a matrix language will be easy, it will not.

7.3. Rashel Rojas

When I found out that I would need to create a compiler to graduate with a degree in Software Systems, I was definitely dreading taking this class. To be honest, I had never really thought about how a compiler worked, or knew what a compiler really was. It was always just there in the background and somehow it let me complete all of my previous coding assignments. But now I can say that I finally understand how it all comes together - but that's not to say that this project wasn't hard. In terms of technical challenges, one of the main challenges for me was understanding how all of the main files worked together to create something as complex as a compiler. What made things more difficult was working with an existing code base. Since our language was very similar to the C language, we began working with the MicroC compiler and modified segments to suit our language. However, some of these files were very long and cluttered with a lot of OCaml code, which, for someone at the start of their OCaml learning experience, required many hours trying to understand what each line was doing, and how that file then related to another file, and so on. This project definitely included a large learning curve, but you can only do as well as the amount of time and effort you put in, which I think we all learned. I'm definitely very thankful to have a team

that was determined to make a functional language. I learned a lot about working on a team, such as the importance of setting deadlines, communicating effectively to update everyone on what was added to our shared repo, and so on.

The lessons I've learned can be advice for future students taking this class. Start early, set weekly deadlines, meet at least once a week with your team and TA. Once you can get the Hello World deliverable working, split up the remaining work such that half of the team works on a subset of features to add and the other half on a different subset of features. This way, there's at least two people working together and because they're working on similar/same features, they can consult each other with questions instead of needing to update the entire team. Also, don't just try to add as many features as you can in one push. Instead, write a few test cases for each new feature that you implement. Furthermore, don't be afraid to reach out to your or other TAs! Be transparent with them about any issues your team is having - they're there to help.

7.4. Mauricio Guerrero

When I found out I had to take a compilers class, I genuinely thought it would be one of the best and worst experiences of my life. Needless to say, it was both. This class as well as Operating Systems are notorious for students talking about how much work they have to do and how it seems like they have so little time to do everything they must do. They were right. Though the course material is rigorous, the satisfaction of understanding how computers work makes it all worth it in the end.

Before this class, I had not idea how compilers worked or the power of functional based programming languages such as Ocaml. I learned that patience and communication is key when understanding difficult concepts such as compilers and functional programming. I also learned that having a consistent schedule with weekly meetings with your team is just as important as starting as soon as you can. Because this semester goes by fast and so much material is covered in every lecture. You should also not be too ambitious when it comes to programming features you want your language to implement. Get the groundwork of your language set first, and if there is enough time, implement the other ideas you had in mind. You should also split work evenly between all of your team members and have one of them be the leader that makes sure everyone else is held accountable for their jobs. Don't be afraid to ask questions to your TA as well. They are there to help you. You also have Professor Edwards himself. You can book an

appointment with him and ask him any questions you and your team might have. This class is worth your time, so give it your best!

8. Appendix

8.1. Project Log

```
commit 10a23f99ce78556d50610b2f423cd52530522523 (HEAD -> main, origin/main,
origin/HEAD)
Merge: 74a5a37 17d9d6d
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Wed Dec 22 18:55:18 2021 -0500

    Merge branch 'main' of https://github.com/54aaron/MX

commit 74a5a3762d29f0ec1b903cc15d679eb7351c67b9
Author: Rashel Rojas <rdr2139@columbia.edu>
:...skipping...
commit 10a23f99ce78556d50610b2f423cd52530522523 (HEAD -> main, origin/main,
origin/HEAD)
Merge: 74a5a37 17d9d6d
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Wed Dec 22 18:55:18 2021 -0500

    Merge branch 'main' of https://github.com/54aaron/MX

commit 74a5a3762d29f0ec1b903cc15d679eb7351c67b9
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Wed Dec 22 18:55:00 2021 -0500

    finished adding more test cases

commit 17d9d6da83dc20d78655c0422c343041f63f5e34
Author: 54aaron <arj2145@columbia.edu>
Date:   Wed Dec 22 18:27:19 2021 -0500

    cleaned up LRM

commit 578dd2fcead25b122aaeac3c7520be9edda508ca
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Wed Dec 22 17:50:22 2021 -0500
```

```
    removed comment from parser

commit aa4d677528eb7a66c4ec324bd5c1c700fb007608
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Wed Dec 22 11:01:56 2021 -0500
:...skipping...
commit 10a23f99ce78556d50610b2f423cd52530522523 (HEAD -> main, origin/main,
origin/HEAD)
Merge: 74a5a37 17d9d6d
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Wed Dec 22 18:55:18 2021 -0500

    Merge branch 'main' of https://github.com/54aaron/MX

commit 74a5a3762d29f0ec1b903cc15d679eb7351c67b9
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Wed Dec 22 18:55:00 2021 -0500

    finished adding more test cases

commit 17d9d6da83dc20d78655c0422c343041f63f5e34
Author: 54aaron <arj2145@columbia.edu>
Date:   Wed Dec 22 18:27:19 2021 -0500

    cleaned up LRM

commit 578dd2fcead25b122aaeac3c7520be9edda508ca
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Wed Dec 22 17:50:22 2021 -0500

    removed comment from parser

commit aa4d677528eb7a66c4ec324bd5c1c700fb007608
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Wed Dec 22 11:01:56 2021 -0500

    fixed formatting for printing matrices

commit 2660993efc7e1816b75102ec422ecea7605a2755
Author: mg4145 <mg4145@columbia.edu>
Date:   Tue Dec 21 20:05:54 2021 -0500
```

```
    Renamed transpose files and fixed order in transformation files

commit f90e3edc7ee366ce97f188a55896e103cd7c612b
Author: mg4145 <mg4145@columbia.edu>
Date:   Tue Dec 21 20:00:59 2021 -0500

    Wrote test files for rotations and transformations

commit 9fb1eb59beb82ddeb1d36c8157a49d4782b384f0
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Tue Dec 21 13:36:49 2021 -0500

    removed unnecessary test files from MX folder

commit 14fe86d95d1312e3781b880c4de43b4f027b89ac
Author: 54aaron <arj2145@columbia.edu>
Date:   Tue Dec 21 11:38:44 2021 -0500

     demo done

commit f5561e955735ebd0f032900cfb47dfdcba46b4c6
Merge: 03f7c1e 571e09e
Author: 54aaron <arj2145@columbia.edu>
Date:   Tue Dec 21 10:32:26 2021 -0500

    Merge branch 'main' of https://github.com/54aaron/MX into main

commit 571e09e9e243879cd773f842007410f37ff1730f
Merge: 4f7ba4d 1d697ea
Author: mg4145 <mg4145@columbia.edu>
Date:   Tue Dec 21 02:18:12 2021 -0500

    Merge branch 'main' of https://github.com/54aaron/MX

commit 4f7ba4dd6717c40a210b3f8740d8dc85d2a2c5d1
Author: mg4145 <mg4145@columbia.edu>
Date:   Tue Dec 21 02:17:31 2021 -0500

    Wrote files mx and out files for while loops

commit 1d697eae81c1690620ec7bb78cd02aa9e25d2ac8
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Tue Dec 21 02:13:33 2021 -0500
```

```
    added non-matrix-related test cases

commit 00351be7a1731df2e86694f6baa397755d85240d
Author: mg4145 <mg4145@columbia.edu>
Date:   Tue Dec 21 02:06:10 2021 -0500

    Wrote mx/out files for for loops testing

commit 7e42cbc734e58b035d046190fc41dbcc7749394b
Author: mg4145 <mg4145@columbia.edu>
Date:   Tue Dec 21 01:52:26 2021 -0500

    Wrote mx and out files for test-if

commit 03f7c1e7b1bb48cc0ee7870f32bb242e7fd18d46
Author: 54aaron <arj2145@columbia.edu>
Date:   Tue Dec 21 01:33:49 2021 -0500

    Demo file finished - might iterate on it later

commit 35fca03b5657de8cfabdeaf60c611fbd831a3c50
Merge: 3e8e3e7 f4cb821
Author: 54aaron <arj2145@columbia.edu>
Date:   Tue Dec 21 00:52:45 2021 -0500

    Merge branch 'main' of https://github.com/54aaron/MX into main

commit f4cb821329fb464404f5716079abf5132c292ff3
Merge: e6fdda2 a63a5c1
Author: mg4145 <mg4145@columbia.edu>
Date:   Tue Dec 21 00:49:21 2021 -0500

    Merge branch 'main' of https://github.com/54aaron/MX

commit e6fdda23bbb91cd307e05ea9743d048153f1b064
Author: mg4145 <mg4145@columbia.edu>
Date:   Tue Dec 21 00:48:03 2021 -0500

    Created .gitignore file

commit e885eab639863ccb8f1d548185fbeda342d3718e
Author: mg4145 <mg4145@columbia.edu>
```

```
Date:   Tue Dec 21 00:45:59 2021 -0500

    Added mx/out for scalar matrix multiplication

commit 703807004b0b1877e578668736cdcf4f4e1e36e7
Author: mg4145 <mg4145@columbia.edu>
Date:   Tue Dec 21 00:26:20 2021 -0500

    Created mx/out files for matrix identity

commit 8595af70de55711d2164d8e51a2845261f7b083e
Author: mg4145 <mg4145@columbia.edu>
Date:   Tue Dec 21 00:19:57 2021 -0500

    Added files for rows and columns

commit a81134d3166aae1ab609de5a43f058248f5adb3b
Author: mg4145 <mg4145@columbia.edu>
Date:   Tue Dec 21 00:10:03 2021 -0500

    Added transpose matrix tests and out files

commit a63a5c1682cfa0f56cb3c2043f8b5f5f7a0485d4
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Tue Dec 21 00:03:34 2021 -0500

    finished adding all fail cases

commit 1feb34193bb2164b994396fa681997a84477a1d9
Author: mg4145 <mg4145@columbia.edu>
Date:   Mon Dec 20 23:57:28 2021 -0500

    Added second test/out for matrix multiplication

commit 0097262f09b774fa70306e34c61833352816e3cc
Author: mg4145 <mg4145@columbia.edu>
Date:   Mon Dec 20 23:47:42 2021 -0500

    Added test and out files for matrix multiplication

commit e344482934f5a720ca14fbc525efea78233a4941
Author: mg4145 <mg4145@columbia.edu>
Date:   Mon Dec 20 23:25:55 2021 -0500
```

```
    Added mx and out file for matrix sub

commit 4f4763aa49cbab41a5cf36dd2b7f16cd6ba26139
Author: mg4145 <mg4145@columbia.edu>
Date:   Mon Dec 20 23:17:09 2021 -0500

    Wrote dot out file test add matrix file

commit b15f7d0c53d86e3e70b16371f46366667cb24286
Author: mg4145 <mg4145@columbia.edu>
Date:   Mon Dec 20 23:16:53 2021 -0500

    Wrote mx file for adding matrices

commit 3e8e3e7956d18efeed9dd96dcc0a5c75e5547f18
Merge: 5be2711 80e43e1
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 23:05:48 2021 -0500

    Merge branch 'main' of https://github.com/54aaron/MX into main

commit 5be27114f0f23a4da5c87363eb07755e16f2ccde
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 23:05:40 2021 -0500

    pi function fully integrated

commit b33aabde2ab66ed726a2b84a95d2d2edde10c13b
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 22:52:02 2021 -0500

    pi in semant

commit 1a9d0b85ed117b77a83a0f7546b9bafba482dd7f
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 22:49:44 2021 -0500

    added pi function to codegen

commit 80e43e14141da0e76ed5415d3bd25eec04897d55
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Mon Dec 20 22:48:37 2021 -0500
```

```
    removed unnecessary fail files from MX folder

commit e78f40f67152329085b12dd67b105ec5e14f32d9
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Mon Dec 20 21:20:09 2021 -0500

    added pi

commit 2d5ceb69a9cf547e795e66f0ba101857a171c1fc
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 20:45:04 2021 -0500

    Int to float casting works on basic operators alone

commit 51bd12406c65885671d6f0b28c5e12fdf1711217
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 20:40:39 2021 -0500

    int to float casting working in direction float + int

commit d3e028f552e1e6dcd1ee073d4c75c33c7576f8cc
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 20:35:19 2021 -0500

    working on implicit int to float casting

commit 385cc143dc1b3d33fcc9448e9ae4f8adc496c858
Merge: 00e7570 dbb883c
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 17:53:26 2021 -0500

    Merge branch 'main' of https://github.com/54aaron/MX into main

commit 00e7570c936f5ed54181fb01a1de6c1fe983342a
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 17:53:07 2021 -0500

    aborted adding increment and decrement - would change structure of
language - too late to do that

commit efefa8611fc7fda22134a779cea72d0328bf9317
Author: 54aaron <arj2145@columbia.edu>
```

```
Date:   Mon Dec 20 17:13:18 2021 -0500

    added both to semant

commit 7532e76bbacc57d6fe0cfa294603118e66a86668
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 16:55:24 2021 -0500

    working on increment and decrement, added them to Scanner

commit dbb883c05d8c6cd1fb556fbf9dab9f045c698993
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Mon Dec 20 16:52:09 2021 -0500

    finished added fail cases for matrix functions

commit b7da38a490ca82a7ec0aebfea742ed0c30406ccf
Merge: 1f6b115 cc5a012
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 16:43:22 2021 -0500

    Merge branch 'main' of https://github.com/54aaron/MX into main

commit 1f6b11505ad050f42572d8d5803045977de25a21
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 16:43:18 2021 -0500

    all assignment operators working nothing broke

commit cc5a012beb2246f46c34e0004f8c72bd005566b2
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Mon Dec 20 16:31:40 2021 -0500

    fixed small formatting error in one test file

commit c1f62f07afe60d9af9d384ef70adb9c0c0523d8c
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 16:31:01 2021 -0500

    minus assign and times assign added to codegen and semant. removed
divide assign

commit 184176fd10aed6defdee8805f9c2f29e7800938c
```

```
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Mon Dec 20 16:28:35 2021 -0500

    fixed small syntax error in mx.c

commit c65ae72e2c02a893fa4b6d5048be38cf5a2af981
Merge: 7ef3e61 179d846
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Mon Dec 20 16:26:16 2021 -0500

    Merge branch 'main' of https://github.com/54aaron/MX

commit 7ef3e61a358204e7b1b6f59baf4f2d6ac1e3a24d
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Mon Dec 20 16:25:52 2021 -0500

    added failure test files for mxmult, scalar multiplication,
transformations, transpose, identity, and printing of matrices

commit ab22b4a4f971696a74881549bb1fe61b5a726505
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 16:25:36 2021 -0500

    added minusassign and timesassign to parser and ast

commit 11b7a3009a437dbf68ed5df926658adb9e907d46
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 16:16:33 2021 -0500

    Assignment operator - Plus assign done

commit 179d846071d6766e90e43b042bd45c940935c3c5
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Mon Dec 20 14:27:20 2021 -0500

    formatting

commit f26d3571f2e873233008b09bc3d9abc219b4f0f0
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Mon Dec 20 14:22:57 2021 -0500

    edited transform comments
```

```
commit 01c54496994ad2faed30cd8d1ec6b0e4aea2a5fc
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 02:34:44 2021 -0500

    make clean

commit bed096a68cf892ec30f6f7c5e61b6878d693ee87
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 02:34:18 2021 -0500

    all prints outputs will have newline now

commit aeed0545c0bad37e981be467f358ddcd4e911288
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 02:32:48 2021 -0500

    num rows implemented

commit a22cf67398d4e0b3675cda96f251ac72f79b3367
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Dec 20 02:29:55 2021 -0500

    numcols implemented

commit 3f54315efbec1e846778f6b6ab1928112f68ff40
Author: 54aaron <arj2145@columbia.edu>
Date:   Sun Dec 19 12:38:00 2021 -0500

    strings now print with new line, working on error checking for matrix
ops

commit 79a05664ecea8bfe0ce25115ab793c9483570c7b
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Sun Dec 19 02:13:00 2021 -0500

    added failure test files for matrix subtraction

commit a7b389f08ede6e14c44a24ba1a221751f2e133eb
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Sun Dec 19 02:00:49 2021 -0500

    added more failure test cases
```

```
commit 7770c4f0ac3f7f2e0592d35f920fd420069f8c75
Author: mg4145 <mg4145@columbia.edu>
Date:   Sat Dec 18 23:12:18 2021 -0500

    Added missing semicolon, everything works again.

commit a54e8b5d4b3e770bf310abc0b983fb894d678103
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Sat Dec 18 21:20:03 2021 -0500

    added numrows & numcols

commit 72e3a2c26320e4e345d8ec5435f68403e4542286
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Sat Dec 18 20:48:44 2021 -0500

    added some failure test cases

commit 69ea69a814a1672b59f250c5519112b011039901
Author: 54aaron <arj2145@columbia.edu>
Date:   Sat Dec 18 14:50:19 2021 -0500

    renamed it

commit 4b66fbd33a337b42ad1502188e411b9723710909
Author: 54aaron <arj2145@columbia.edu>
Date:   Sat Dec 18 14:49:43 2021 -0500

    it works :<

commit 85e9b0cb5234063fc77a7605cfb0f8019a222a01
Author: 54aaron <arj2145@columbia.edu>
Date:   Sat Dec 18 14:46:23 2021 -0500

    developing general compilation script

commit 51b0e50c8f67bfe1ec1031a66aa439eab3b1c749
Author: 54aaron <arj2145@columbia.edu>
Date:   Fri Dec 17 22:17:48 2021 -0500

    fixed test-matrix

commit 9ba1cede929f0e7ee2e5aab3acb343c1617c3b17
```

```
Author: 54aaron <arj2145@columbia.edu>
Date:   Fri Dec 17 19:09:05 2021 -0500

    edited matrix file to use die function instead of perror

commit c4d0a9278be0b1101d638c7f1bbe83d332925e50
Author: 54aaron <arj2145@columbia.edu>
Date:   Fri Dec 17 19:07:33 2021 -0500

    transformation function works

commit 242cfd3cdbdd1d78b5b39a7d01ba8958dbe29625
Author: 54aaron <arj2145@columbia.edu>
Date:   Fri Dec 17 18:57:37 2021 -0500

    added transformation to semant

commit 461f53a11c32a9c5ebbb2181cd4f953c7bbfea2d
Author: 54aaron <arj2145@columbia.edu>
Date:   Fri Dec 17 18:56:36 2021 -0500

    added transformation function to codegen

commit d4077855abb902a642a44dbc2db63be751fe03cd
Author: 54aaron <arj2145@columbia.edu>
Date:   Fri Dec 17 18:52:43 2021 -0500

    fixed bugs in C library

commit 84797d11b95fd3c45333439ddc53e3eee810c1ea
Merge: b948597 543b72d
Author: 54aaron <arj2145@columbia.edu>
Date:   Fri Dec 17 18:42:34 2021 -0500

    still doing git pull

commit b9485979f77fdeadd661dd855032cd2708e24c47
Author: 54aaron <arj2145@columbia.edu>
Date:   Fri Dec 17 18:41:00 2021 -0500

    doing a git pull

commit 543b72d60a5257134d761f84060bfd67e41d2ae4
```

```
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Fri Dec 17 16:45:18 2021 -0500

    edited transformation function

commit 021a2c4a345c8ec14d56ff634e24a434b8f25f05
Author: mg4145 <mg4145@columbia.edu>
Date:   Fri Dec 17 14:19:13 2021 -0500

    Updated test-matrix dot mx and dot out files.

commit 0d37f1abe6c93e6455c329488e4d586202c8bfec
Author: 54aaron <arj2145@columbia.edu>
Date:   Thu Dec 16 23:44:14 2021 -0500

    cleaned up print statements from C file

commit c0560aac4c4a3ffbb177fde5226297de84fbe2f7
Author: 54aaron <arj2145@columbia.edu>
Date:   Thu Dec 16 23:39:05 2021 -0500

    did a make clean

commit 08c18c043705981817d0d7ab30439c73ea6fe9dd
Author: 54aaron <arj2145@columbia.edu>
Date:   Thu Dec 16 23:38:36 2021 -0500

    Built in functions can now have multiple arguments

commit a911ce04953f818f63311aaa743ef8dc45c1052c
Author: mg4145 <mg4145@columbia.edu>
Date:   Thu Dec 16 22:12:26 2021 -0500

    Moved .out files to tests

commit 4879135f412dff61647287effb07972b223eb51f
Author: mg4145 <mg4145@columbia.edu>
Date:   Thu Dec 16 22:11:45 2021 -0500

    Created fail file for out language mx

commit deea351ed8a27172ec5aa3958180d346d3c019c3
Author: mg4145 <mg4145@columbia.edu>
```

```
Date:   Thu Dec 16 22:10:56 2021 -0500

    Added .err file for corresponding fail file

commit 1a44cf6420ac9f87b9088e300c707ba930041dfb
Author: mg4145 <mg4145@columbia.edu>
Date:   Thu Dec 16 22:10:07 2021 -0500

    Added PHONY test variable

commit 87144cb23a76f89d17d90b0a85f9ba791796b22e
Author: mg4145 <mg4145@columbia.edu>
Date:   Thu Dec 16 21:53:34 2021 -0500

    Added test-matrix out file

commit aef6f6fe0ee988f1b55b037373aa432a8b17cbcc
Author: mg4145 <mg4145@columbia.edu>
Date:   Thu Dec 16 21:53:14 2021 -0500

    Added test-hello file

commit 9883ee6da69c83e9fd3e11e238eaab87041357e1
Author: mg4145 <mg4145@columbia.edu>
Date:   Thu Dec 16 21:52:48 2021 -0500

    Added test-gcd file

commit 001bac87d237ca28ad5232950ef20af26ba781b9
Author: mg4145 <mg4145@columbia.edu>
Date:   Thu Dec 16 21:51:21 2021 -0500

    Updated file to newest changes

commit 8e650455fa136293bca2ac6536748c6cc626e814
Author: mg4145 <mg4145@columbia.edu>
Date:   Thu Dec 16 21:16:42 2021 -0500

    Added -lm flag for mx.c compilation.

commit e637dfaee4aa45a4e60cf42886810bba811be099
Author: 54aaron <arj2145@columbia.edu>
Date:   Thu Dec 16 21:11:41 2021 -0500
```

```
    made scalar operator. No warnings woohoo!

commit ed206012bd701f1d420a1f7b7a71e2c5a90c406c
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Thu Dec 16 21:00:15 2021 -0500

    Delete matrix.c

commit 1a0ff7a09631b4a43f053b89f53ca2e589d2f98a
Merge: 4653717 f4dc6de
Author: 54aaron <arj2145@columbia.edu>
Date:   Thu Dec 16 20:54:26 2021 -0500

    just did a pull

commit 46537178296c0830418beb1f4552e5e1f682d4eb
Author: 54aaron <arj2145@columbia.edu>
Date:   Thu Dec 16 20:53:40 2021 -0500

    identity up n running as a function

commit f4dc6de9574e4fd70cbca6ea859525073b6954bd
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Thu Dec 16 20:53:05 2021 -0500

    Update mx.c

commit c6f2fa0670a4693958985ad2978736ebc02447ef
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Thu Dec 16 20:50:24 2021 -0500

    update ID

commit f2835ce2fc677d4afb6fb19a835d2d58e3a87656
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Thu Dec 16 20:14:43 2021 -0500

    Update mx.c

commit 489b133172914003eb78f8d3e221b4a4f6ffdb1c
Merge: 3f3a981 2b76e86
Author: 54aaron <arj2145@columbia.edu>
```

```
Date:   Thu Dec 16 20:14:08 2021 -0500

    Merge branch 'main' of https://github.com/54aaron/MX into main

commit 3f3a9811899e2f3261b5030ec49129713924f82c
Author: 54aaron <arj2145@columbia.edu>
Date:   Thu Dec 16 20:14:05 2021 -0500

    going to pull

commit 2b76e86904401533f688b3557771053762964a03
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Thu Dec 16 19:42:18 2021 -0500

    updated rotation

commit 74cadcc90db8148ddfce4826ce9a99690caec697
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Thu Dec 16 19:35:01 2021 -0500

    added rotation matrix

commit 1987f2ec6f2db146d3f3e10978670fc8da7905b1
Merge: a9b2d68 8a68d10
Author: 54aaron <arj2145@columbia.edu>
Date:   Thu Dec 16 19:28:35 2021 -0500

    Merge branch 'main' of https://github.com/54aaron/MX into main

commit 8a68d10a61d37abaadab8603938baf2ec47d62cd
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Thu Dec 16 17:49:32 2021 -0500

    Update mx.c

commit e8463c5128a4b2cb04c617bf3f44bd448b896237
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Thu Dec 16 17:44:59 2021 -0500

    updated display, added identity & scalar mult

commit a9b2d68f98554f1d1c5d933b97929e1758046e84
Author: 54aaron <arj2145@columbia.edu>
```

```
Date:   Thu Dec 16 16:37:26 2021 -0500

    got rid of a bunch of warnings and stuff

commit 9e7ddea01d1cd377876e718719353dfa86b19f3a
Author: 54aaron <arj2145@columbia.edu>
Date:   Thu Dec 16 13:01:28 2021 -0500

    Mxsub working as an operator

commit bd501df8d3961724d3b232f5e1018a632c63ea6b
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Thu Dec 16 12:26:49 2021 -0500

    added sub & mult

commit 1849d352ec5ea721fcb7b19205719470d63efbb5
Author: 54aaron <arj2145@columbia.edu>
Date:   Wed Dec 15 19:32:05 2021 -0500

    Matrix addition working

commit 5e7746c43fe096fbf1061724f9503403367158b7
Author: 54aaron <arj2145@columbia.edu>
Date:   Wed Dec 15 16:51:21 2021 -0500

    Transpose is both a unary operator and also a function

commit 73cfec35904e053e3bea54fbc2d5f3ad8c5089f1
Merge: 45a8b90 58ca749
Author: 54aaron <arj2145@columbia.edu>
Date:   Wed Dec 15 15:44:18 2021 -0500

    did a git pull

commit 45a8b90416ab43cc470f9b7d6e8bde4931caaa60
Author: 54aaron <arj2145@columbia.edu>
Date:   Wed Dec 15 15:41:52 2021 -0500

    Transpose working as a function - will try to get working as a unary
operator (LMAOOOOOOOOOOOO)

commit fbf931f9f4bad28c2ad57f3afba54f18415e2df5
```

```
Author: 54aaron <arj2145@columbia.edu>
Date:   Wed Dec 15 13:09:41 2021 -0500

    fixed row and column mix up

commit 58ca74963bb8a250e9e7ed064216b721f0415434
Author: Aaron Jackson <77600293+54aaron@users.noreply.github.com>
Date:   Mon Dec 13 22:18:16 2021 -0500

    Update mx.c

commit 882d8fcf982e68cd06c392d558ab8bb29f2f394f
Author: Aaron Jackson <77600293+54aaron@users.noreply.github.com>
Date:   Mon Dec 13 22:08:50 2021 -0500

    Update mx.c

commit ff2f09a09c7a967db06d1f92292b68051d1d1be1
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Mon Dec 13 22:06:17 2021 -0500

    updated

commit 3fd43725def1df0e182b1e7fb15150afdd1cef49
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Mon Dec 13 18:54:40 2021 -0500

    printmatrix added

commit 0de668a5161140d2a0dd7ed98830a50d341805c5
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Mon Dec 13 18:35:47 2021 -0500

    added mxscale

commit 531dc8802241f6faaeb82f416642be6b40134b54
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Mon Dec 13 17:10:27 2021 -0500

    Create matrix.c

commit 47df1faec04ba1b3c880cfcc84ef5b395ce19474
Author: 54aaron <arj2145@columbia.edu>
```

```
Date:   Sun Dec 5 22:46:43 2021 -0500

    Matrix init and matrix store working (I think) changed Semant to get
rows and columns from matrix literal changed SMx to take tuple of elements,
rows, and columns, changed parser to reverse elems_list and rows_list and
added some more functions to the library and codegen

commit 7a802adc18921bd80664cc5baec24b0003b558d8
Author: 54aaron <arj2145@columbia.edu>
Date:   Fri Dec 3 20:04:05 2021 -0500

    fixed make clean

commit e90116f273ba110098e08bd7be337a757b0f484e
Merge: 5af68fa ace86dd
Author: Aaron Jackson <77600293+54aaron@users.noreply.github.com>
Date:   Fri Dec 3 19:58:45 2021 -0500

    Merge pull request #2 from 54aaron/experiment

    Experiment

commit ace86ddb8b6633eb63c56a5963e226c368b2c746 (origin/experiment)
Author: 54aaron <arj2145@columbia.edu>
Date:   Fri Dec 3 19:53:25 2021 -0500

    codegen pipeline working uhh made script thats all folks

commit 8589a34d4854a41f7572e34309e36a56444ccca9
Author: 54aaron <arj2145@columbia.edu>
Date:   Tue Nov 30 22:39:59 2021 -0500

    mx pipeline functioning

commit cc85e818f06537a5ff2e872668af4b1e6f8447bb
Author: 54aaron <arj2145@columbia.edu>
Date:   Tue Nov 30 13:57:57 2021 -0500

    Got C file to link, added Mx case to semant case matching on line 95 we
will need to change this, added SMx of int list list on sast line 9 case
matching

commit e400a71ac9861bb454c72be5709b623318cdd852
```

```
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Nov 29 22:08:46 2021 -0500

    Working on stuff

commit 5af68fa12dcb3a5812c23722983de32949eed2aa
Author: 54aaron <arj2145@columbia.edu>
Date:   Fri Nov 26 16:37:16 2021 -0500

    Matrix declaration editing in parser

commit 9cefabe032133b12c3334f0fc8ec0efe830a2d12
Author: wo2168 <71407594+wo2168@users.noreply.github.com>
Date:   Tue Nov 16 19:38:23 2021 -0800

    updated README to include compilation instructions

commit b30c41cf5d5e49b893d0b8bb5cf0115a2a3a847a
Author: 54aaron <arj2145@columbia.edu>
Date:   Tue Nov 16 21:37:54 2021 -0500

    Print String Support

commit 998179276cebe47f5290ace9de08494022fc8765
Merge: 5bb8289 87f6647
Author: Aaron Jackson <77600293+54aaron@users.noreply.github.com>
Date:   Tue Nov 16 21:19:30 2021 -0500

    Merge pull request #1 from 54aaron/hello_world_deliverable

    Hello world deliverable

commit 87f6647de041d4d8f07caee14b48e9075e58d692
(origin/hello_world_deliverable)
Author: 54aaron <arj2145@columbia.edu>
Date:   Thu Nov 11 21:12:17 2021 -0500

    included bash, deliverable complete

commit 9e5c48736b7a5fd9ee21d922a0c12644bb879d8f (origin/hello_world)
Author: 54aaron <arj2145@columbia.edu>
Date:   Wed Nov 10 23:37:48 2021 -0500
```

```
    gcd works!

commit ee27b8ea41c7596e65f45f5c555305bb8448ccb8
Author: 54aaron <arj2145@columbia.edu>
Date:   Wed Nov 10 21:25:46 2021 -0500

    new stuff

commit 5bb8289b952a4e1a23d2f265ceae7f49c2bc9140
Author: 54aaron <arj2145@columbia.edu>
Date:   Mon Nov 8 22:02:14 2021 -0500

    ast sast

commit 0364aca7b9371cc67e929bc3a3fa167375c47f15
Author: 54aaron <arj2145@columbia.edu>
Date:   Sat Nov 6 19:44:52 2021 -0400

    changed transpose

commit fd5d7d712bc2613920b20fb322362e8c600cd7a9
Author: 54aaron <arj2145@columbia.edu>
Date:   Sat Nov 6 19:42:25 2021 -0400

    changed more stuff

commit 75bf6b2273e02d2de3d1897bfe6a78f9f2a17f7b
Merge: 4ae1c9c 9004838
Author: 54aaron <arj2145@columbia.edu>
Date:   Sat Nov 6 19:39:32 2021 -0400

    Merge branch 'main' of https://github.com/54aaron/MX into main

commit 4ae1c9c7083fd0309f20b8bfd2d649ea4cf46d84
Author: 54aaron <arj2145@columbia.edu>
Date:   Sat Nov 6 19:38:32 2021 -0400

    Scanner complete

commit 90048382ca19957534607cd8c9b64a1b43bc1e0a
Author: Mg4145 <65383117+Mg4145@users.noreply.github.com>
Date:   Fri Nov 5 16:23:29 2021 -0400
```

```
    Delete ast.ml

    Wrong file extension. It should have been dot mli and not dot ml

commit 84866c99934e598b448c7ef05ee323d3e046cd30
Author: mg4145 <mg4145@columbia.edu>
Date:   Fri Nov 5 16:21:46 2021 -0400

    Gave correct file extension/

commit 28471568bd53273675d940afd4958a568d12286e
Author: 54aaron <arj2145@columbia.edu>
Date:   Fri Nov 5 16:03:00 2021 -0400

    Parser complete

commit 111f1ae12c006d261240dda46455e6ee524c360d
Author: 54aaron <arj2145@columbia.edu>
Date:   Wed Nov 3 22:07:19 2021 -0400

    more parser work

commit 186fd1a6e6040169617e62505ed9eb1daa894bb3
Author: 54aaron <arj2145@columbia.edu>
Date:   Wed Nov 3 21:25:43 2021 -0400

    working on declaration

commit d33a954481ced4767880a27e2af172247ef5137c
Author: Rashel Rojas <rdr2139@columbia.edu>
Date:   Sun Oct 31 14:15:04 2021 -0400

    Testing by Rashel

commit ea0353e8365f96b8e6f6cd2fe28f5b6d2285463e
Author: wo2168 <wo2168@columbia.edu>
Date:   Sun Oct 31 11:06:52 2021 -0700

    test

commit cc01438b826c8dea52e1d33a93439f7409a48f32
Author: 54aaron <arj2145@columbia.edu>
Date:   Sun Oct 31 13:43:04 2021 -0400
```

```
    Added Matrix declarations, operations, and tokens to parser

commit e9d0c93373bccd62a66cb186ff7d701e8176f989
Author: mg4145 <mg4145@columbia.edu>
Date:   Sat Oct 30 00:49:44 2021 -0400

    Included team members names

commit ba76f8ecc64ec34574b70cd374dfdf3891a4e6f2
Author: mg4145 <mg4145@columbia.edu>
Date:   Sat Oct 30 00:44:00 2021 -0400

    Added MAKEFILE. Subject to change.

commit 82c0fc8dae1915e7f68f03b1579924604b1ecd17
Author: mg4145 <mg4145@columbia.edu>
Date:   Sat Oct 30 00:35:03 2021 -0400

    Ast template. Subject to change.

commit 17a3eede5517d7b99dee3a3e7a455cda64990dfc
Author: mg4145 <mg4145@columbia.edu>
Date:   Sat Oct 30 00:31:10 2021 -0400

    Added scanner ex from MicroC. Must be changed.

commit bf534bf67dadb59e987265a3c31024fa6ca82d09
Author: mg4145 <mg4145@columbia.edu>
Date:   Sat Oct 30 00:19:15 2021 -0400

    Created README and added basic information.

commit 83a6302831fb6320131695e0fa6e824a3a1e847f
Author: mg4145 <mg4145@columbia.edu>
Date:   Sat Oct 30 00:10:33 2021 -0400

    Renamed parser.mly.txt to parser.mly

commit f72e395c414a1a03d055a98f180d7c2503a5740a
Author: mg4145 <mg4145@columbia.edu>
Date:   Fri Oct 29 21:21:10 2021 -0400
```

```
    Included parserl.mly file

commit f39ba86a5f68d7a25fdc22d6f11d139da2b46753
Author: Aaron Jackson <77600293+54aaron@users.noreply.github.com>
Date:   Fri Oct 29 21:06:17 2021 -0400

    Add files via upload
(END)
```

8.2. References

This project refers to previous Matrix language projects Matrx (Fall 2018) and XIRTAM (Spring 2021) for guidelines to construct our own language. The MX language is also built on top of Micro C.

8.3. Project Code

8.3.1. Scanner.mll (Rashel Rojas, Wilderness Oberman, Aaron Jackson, Mauricio Guerrero)

```
{ open Parser }



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

rule token = parse
        [' ' '\t' '\r' '\n']    { token lexbuf }
    | "/*"                  { comment lexbuf }
    | "#"                   { singlecomment lexbuf }

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

    | "if"                  { IF }
    | "else"                { ELSE }
```

```
  | "while"              { WHILE }
  | "for"                { FOR }

  | '='                  { ASSIGN }
  | '+'                  { PLUS }
  | '-'                  { MINUS }
  | '*'                  { TIMES }
  | '/'                  { DIVIDE }
  | "+="                 { PLUSASSIGN }
  | "-="                 { MINUSASSIGN }
  | "*="                 { TIMESASSIGN }

  | "+."                 { MXPLUS }
  | "-."                 { MXMINUS }
  | "*."                 { MXMX }
  | "**."                { MXSCALE }
  | '''                  { TRANSPOSE } (* ASK TA - CHANGE TO ^ *)

  | "=="                 { EQ }
  | ">"                  { GT }
  | '<'                  { LT }
  | "<="                 { LEQ }
  | "!="                 { NEQ }
  | ">="                 { GEQ }
  | "!"                  { NOT }

  | "||"                 { OR }
  | "&&"                 { AND }

  | ';'                  { SEMI }
  | ','                  { COMMA }

  | "int"                { INT }
  | "Matrix"             { MATRIX }
  | "String"             { STRING }
  | "bool"               { BOOL }
  | "true"               { BLIT(true) }
  | "false"              { BLIT(false) }
  | "float"              { FLOAT }
  | "void"               { VOID }
  | "return"             { RETURN }
  | "null"               { NULL }
```

```
    | '"' ([^ '"']* as lit) '"' { STRINGLIT(lit) }

    | digit+ as lxm                                                  {
LITERAL(int_of_string lxm) }
    | digit+ '.' digit* (['e' 'E'] ['+' '-']? digits)? as lxm        {
FLIT(lxm) }
    | ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']* as lxm         {
ID(lxm) }
    | eof                                                            { EOF }
    | _ as ch                                                        { raise
(Failure("illegal character " ^ Char.escaped ch)) }

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

and singlecomment = parse
    "\n" { token lexbuf }
    | _ { singlecomment lexbuf }
```

8.3.2. Parser.mly (Rashel Rojas, Wilderness Oberman, Aaron Jackson, Mauricio Guerrero)

```
%{ open Ast %}

%token LPAREN RPAREN LBRACE RBRACE LBRACKET RBRACKET SEMI COMMA TRANSPOSE
%token PLUS MINUS TIMES DIVIDE ASSIGN EQ PLUSASSIGN MINUSASSIGN TIMESASSIGN
%token IF ELSE WHILE FOR NOT NOELSE
%token INT BOOL FLOAT STRING RETURN MATRIX VOID NULL
%token NEQ LT GT LEQ GEQ AND OR
%token MXPLUS MXMINUS MXMX MXSCALE
%token <int> LITERAL
%token <string> ID FLIT
%token <bool> BLIT
%token <string> STRINGLIT
%token EOF

%nonassoc NOELSE
%nonassoc ELSE
```

```
%right ASSIGN PLUSASSIGN MINUSASSIGN TIMESASSIGN DIVIDEASSIGN
%left OR
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS MXPLUS MXMINUS
%left TIMES DIVIDE MXMX MXSCALE
%right NOT
%left TRANSPOSE

%start program
%type <Ast.program> program

%%
program: decls EOF { $1 }

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

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

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

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

typ:
      INT           { Int }
    | BOOL          { Bool }
    | STRING        { String }
    | FLOAT         { Float  }
    | MATRIX        { Matrix(Int) }
    | VOID          { Void }

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

vdecl:
```

```
          typ ID SEMI                                { ($1, $2) }
     /*   | typ ID ASSIGN expr SEMI                    { ($1,$2,
Assign($2,$4))}
        | INT   typ ID SEMI                           { ($2, $3) }   int
matrix m;
        | FLOAT typ ID SEMI                           { ($2, $3) }  */

matrix_literal:
          LBRACKET row_list RBRACKET                                  {
List.rev $2 }            /* ASK TA */

row_list:
          /* nothing */                                               {
[] }
          | LBRACKET elems_list RBRACKET                              {
[List.rev $2] }      /* Matrix m = [[1,2,3]] */
          | row_list COMMA LBRACKET elems_list RBRACKET               {
(List.rev $4)::$1 }    /* Matrix m = [[1,2,3],[4,5,6],[7,8,9]] */

/* Matrix m = [,[1,2,3]] */

elems_list:
            LITERAL                                    { [$1] }
          | elems_list COMMA LITERAL                   { $3::$1 }  /* ASK TA
*/

expr:
    LITERAL                     { Literal($1) }
    | FLIT                      { Fliteral($1) }
    | BLIT                      { BoolLit($1) }
    | ID                        { Id($1) }
    | STRINGLIT                 { Stringlit($1) }
    | matrix_literal            { Mx($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 MXPLUS expr          { Binop( $1, Mxadd,$3) }
    | expr MXMINUS expr         { Binop( $1, Mxsub, $3) }
    | expr MXMX expr            { Binop( $1, Mxtimes, $3) }
    | expr MXSCALE expr         { Binop( $1, Mxscale, $3) }
    | expr TRANSPOSE            { Unop( Transpose, $1) } /* our transpose
operation */
```

```
    | expr EQ expr               { Binop($1, Equal, $3) }
    | expr NEQ expr              { Binop($1, Neq, $3) }
    | expr LT expr               { Binop($1, Less, $3) }
    | expr LEQ expr              { Binop($1, Leq, $3) }
    | expr GT expr               { Binop($1, Greater, $3) }
    | expr GEQ expr              { Binop($1, Geq, $3) }
    | expr AND expr              { Binop($1, And, $3) }
    | expr OR expr               { Binop($1, Or, $3) }
    | MINUS expr %prec NOT       { Unop(Neg, $2) }      /*   Ask TA about
this */
    | NOT expr                   { Unop(Not, $2) }
    | ID ASSIGN expr             { Assign($1, $3) }
    | ID PLUSASSIGN expr         { Plusassign($1, $3) }
    | ID MINUSASSIGN expr        { Minusassign($1, $3) }
    | ID TIMESASSIGN expr        { Timesassign($1, $3) }
    | ID LPAREN args_opt RPAREN  { Call($1, $3) }
    | LPAREN expr RPAREN         { $2 }

/* Matrix m = [[1,2,3],[4,5,6]] */

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

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

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

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

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

8.3.3. Ast.ml (Rashel Rojas, Wilderness Oberman, Aaron Jackson, Mauricio Guerrero)

```
type op = Add | Sub | Mult | Div | Equal | Neq | Less | Leq
            | Greater | Geq | And | Or | Mxadd | Mxsub | Mxtimes |
Mxscale

type uop = Neg | Not | Transpose

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

type bind = typ * string

type expr = Literal of int | Fliteral of string | BoolLit of bool
            | Id of string
            | Stringlit of string
            | Mx of int list list
            | Binop of expr * op * expr | Unop of uop * expr
            | Assign of string * expr
            | Plusassign of string * expr
            | Minusassign of string * expr
            | Timesassign of string * expr
            | Call of string * expr list
            | Noexpr

type stmt = Block of stmt list
            | Expr of expr
            | Return of expr
            | If of expr * stmt * stmt
            | For of expr * expr * expr * stmt
            | While of expr * stmt
```

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

type program = bind list * func_decl list

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

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

let rec string_of_expr = function
    Literal(l) -> string_of_int l
  | Fliteral(l) -> l
  | Stringlit(l) -> l
  | BoolLit(true) -> "true"
  | BoolLit(false) -> "false"
  | Mx(l) -> let flat_list = List.flatten l in "\n[" ^ String.concat
```

```
"," (List.map string_of_int flat_list) ^ "]\n"   (* WE SHOULD
PROBABLY COME BACK TO THIS *)
  | Id(s) -> s
  | Binop(e1, o, e2) ->
      string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr
e2
  | Unop(o, e) -> string_of_uop o ^ string_of_expr e
  | Assign(v, e) -> v ^ " = " ^ string_of_expr e
  | Plusassign(v, e) -> v ^ " += " ^ string_of_expr e
  | Minusassign(v, e) -> v ^ " -= " ^ string_of_expr e
  | Timesassign(v, e) -> v ^ " *= " ^ string_of_expr e
  | Call(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | Noexpr -> ""

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

let string_of_typ = function
    Int -> "int"
  | Bool -> "bool"
  | Float -> "float"
  | Void -> "void"
  | String -> "string"
  | Matrix _ -> "matrix"
```

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

let string_of_fdecl fdecl =
  string_of_typ fdecl.typ ^ " " ^
  fdecl.fname ^ "(" ^ String.concat ", " (List.map snd fdecl.formals)
^
  ")\n{\n" ^
  String.concat "" (List.map string_of_vdecl fdecl.locals) ^
  String.concat "" (List.map string_of_stmt fdecl.body) ^
  "}\n"

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

8.3.4. semant.ml (Aaron Jackson)

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

open Ast
open Sast

module StringMap = Map.Make(String)

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

   Check each global variable, then check each function *)

let check (globals, functions) =

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

```
      raise (Failure ("duplicate " ^ kind ^ " " ^ n1))
    | _ :: t -> dups t
  in dups (List.sort (fun (_,a) (_,b) -> compare a b) binds)
in

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

check_binds "global" globals;

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

(* Collect function declarations for built-in functions: no bodies *)
let built_in_decls =
  let add_bind map (name, ty, ret) = StringMap.add name {
    typ = ret;
    fname = name;
    formals = (* create list of args *)
        (
          let rec bind_funcs = (function
            [] -> []
            | fst::snd -> (fst, "x")::(bind_funcs snd))
          in
          bind_funcs ty
        );
    locals = []; body = [] } map
  in List.fold_left add_bind StringMap.empty [
  ("print", [Int], Void);
  ("printb", [Bool], Void);
  ("printf", [Float], Void);
  ("prints", [String], Void);
  ("printbig", [Int], Void);
  ("transpose", [Matrix(Int)], Matrix(Int));
  ("numCols", [Matrix(Int)], Int);
  ("numRows", [Matrix(Int)], Int);
  ("transformation", [Matrix(Int); Int], Matrix(Int));
  ("identity", [Int], Matrix(Int));
  ("twoFunc", [Int; Int], Void);
  ("pi", [], Float);
  ("print_matrix", [Matrix(Int)], Void)
   ]
in
```

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

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

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

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

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

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

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

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

    (* Return a semantically-checked expression, i.e., with a type *)
    let rec expr = function
        Literal  l -> (Int, SLiteral l)
      | Fliteral l -> (Float, SFliteral l)
      | Stringlit l-> (String, SStringlit l)
      | BoolLit l  -> (Bool, SBoolLit l)
      | Mx l       ->

      (* get number of rows *)
      let rows = List.length l in

      (* infer number of columns from number of elements in first row *)
      let cols = List.length (List.hd l) in

      (* function to check that all rows are the same size and raise an
error if they are not *)
      let col_check list = List.map (fun v -> if List.length v != cols then
raise (Failure "Matrix rows are not all the same length")) list in

      ignore(col_check l); (Matrix(Int), SMx (l, rows, cols))

      (* print_int(cols); (Matrix(Int), SMx l) *)
      | Noexpr     -> (Void, SNoexpr)
      | Id s       -> (type_of_identifier s, SId s)
      | Assign(var, e) as ex ->
          let lt = type_of_identifier var
          and (rt, e') = expr e in
          let err = "illegal assignment " ^ string_of_typ lt ^ " = " ^
            string_of_typ rt ^ " in " ^ string_of_expr ex
          in (check_assign lt rt err, SAssign(var, (rt, e')))
      | Plusassign(var, e) as ex ->
          let lt = type_of_identifier var
          and (rt, e') = expr e in
          let err = "illegal assignment " ^ string_of_typ lt ^ " = " ^
            string_of_typ rt ^ " in " ^ string_of_expr ex
          in (check_assign lt rt err, SPlusassign(var, (rt, e')))
      | Minusassign(var, e) as ex ->
```

```
          let lt = type_of_identifier var
          and (rt, e') = expr e in
          let err = "illegal assignment " ^ string_of_typ lt ^ " = " ^
            string_of_typ rt ^ " in " ^ string_of_expr ex
          in (check_assign lt rt err, SMinusassign(var, (rt, e')))
      | Timesassign(var, e) as ex ->
          let lt = type_of_identifier var
          and (rt, e') = expr e in
          let err = "illegal assignment " ^ string_of_typ lt ^ " = " ^
            string_of_typ rt ^ " in " ^ string_of_expr ex
          in (check_assign lt rt err, STimesassign(var, (rt, e')))
      | Unop(op, e) as ex ->
          let (t, e') = expr e in
          let ty = match op with
            Neg when t = Int || t = Float -> t
          | Not when t = Bool -> Bool
          | Transpose when t = Matrix(Int) -> t
          | _ -> raise (Failure ("illegal unary operator " ^
                                 string_of_uop op ^ string_of_typ t ^
                                 " in " ^ string_of_expr ex))
          in (ty, SUnop(op, (t, e')))
      | Binop(e1, op, e2) as e ->
          let (t1, e1') = expr e1
          and (t2, e2') = expr e2 in
          (* All binary operators require operands of the same type *)
          let same = t1 = t2 in
          (* Determine expression type based on operator and operand types
*)
          let ty = match op with
            Add | Sub | Mult | Div                  when same && t1 = Int
-> Int
          | Add | Sub | Mult | Div                  when same && t1 =
Float -> Float
          | Add | Sub | Mult | Div                  when (t1 = Float && t2
= Int) || (t1 = Int && t2 = Float)  -> Float
          | Mxadd | Mxsub | Mxtimes                 when same && t1 =
Matrix(Int) -> Matrix(Int)
          | Mxscale                                 when t1 = Matrix(Int)
&& t2 = Int -> (Matrix(Int))
          | Equal | Neq            when same               -> Bool
          | Less | Leq | Greater | Geq
                     when same && (t1 = Int || t1 = Float) -> Bool
          | And | Or when same && t1 = Bool -> Bool
```

```
          | _ -> raise (
            Failure ("illegal binary operator " ^
                     string_of_typ t1 ^ " " ^ string_of_op op ^ " " ^
                     string_of_typ t2 ^ " in " ^ string_of_expr e))
          in (ty, SBinop((t1, e1'), op, (t2, e2')))
      | Call(fname, args) as call ->
          let fd = find_func fname in
          let param_length = List.length fd.formals in
          if List.length args != param_length then
            raise (Failure ("expecting " ^ string_of_int param_length ^
                            " arguments in " ^ string_of_expr call))
          else let check_call (ft, _) e =
            let (et, e') = expr e in
            let err = "illegal argument found " ^ string_of_typ et ^
              " expected " ^ string_of_typ ft ^ " in " ^ string_of_expr e
            in (check_assign ft et err, e')
          in
          let args' = List.map2 check_call fd.formals args
          in (fd.typ, SCall(fname, args'))
    in

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

    (* Return a semantically-checked statement i.e. containing sexprs *)
    let rec check_stmt = function
        Expr e -> SExpr (expr e)
      | If(p, b1, b2) -> SIf(check_bool_expr p, check_stmt b1, check_stmt
b2)
      | For(e1, e2, e3, st) ->
        SFor(expr e1, check_bool_expr e2, expr e3, check_stmt st)
      | While(p, s) -> SWhile(check_bool_expr p, check_stmt s)
      | Return e -> let (t, e') = expr e in
        if t = func.typ then SReturn (t, e')
        else raise (
        Failure ("return gives " ^ string_of_typ t ^ " expected " ^
               string_of_typ func.typ ^ " in " ^ string_of_expr e))

          (* A block is correct if each statement is correct and nothing
             follows any Return statement.  Nested blocks are flattened. *)
```

```
      | Block sl ->
          let rec check_stmt_list = function
              [Return _ as s] -> [check_stmt s]
            | Return _ :: _   -> raise (Failure "nothing may follow a
return")
            | Block sl :: ss  -> check_stmt_list (sl @ ss) (* Flatten
blocks *)
            | s :: ss         -> check_stmt s :: check_stmt_list ss
            | []              -> []
          in SBlock(check_stmt_list sl)

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

8.3.5. sast.ml (Aaron Jackson)

```
open Ast

type sexpr = typ * sx

and sx =
    SLiteral of int
| SFliteral of string
| SBoolLit of bool
| SMx of int list list * int * int
| SId of string
| SStringlit of string
| SBinop of sexpr * op * sexpr
| SUnop of uop * sexpr
| SAssign of string * sexpr
| SPlusassign of string * sexpr
| SMinusassign of string * sexpr
```

```
| STimesassign of string * sexpr
| SCall of string * sexpr list
| SNoexpr

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

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

type sprogram = bind list * sfunc_decl list

(* Pretty-printing functions *)

let rec string_of_sexpr (t, e) =
  "(" ^ string_of_typ t ^ " : " ^ (match e with
    SLiteral(l) -> string_of_int l
  | SBoolLit(true) -> "true"
  | SBoolLit(false) -> "false"
  | SStringlit(l) -> l
  | SFliteral(l) -> l
  | SMx(l,_,_) -> let flat_list = List.flatten l in "\n[" ^ String.concat
"," (List.map string_of_int flat_list) ^ "]\n"
  | SId(s) -> s
  | SBinop(e1, o, e2) ->
      string_of_sexpr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_sexpr e2
  | SUnop(o, e) -> string_of_uop o ^ string_of_sexpr e
  | SAssign(v, e) -> v ^ " = " ^ string_of_sexpr e
  | SPlusassign(v, e) -> v ^ " += " ^ string_of_sexpr e
  | SMinusassign(v, e) -> v ^ " -= " ^ string_of_sexpr e
  | STimesassign(v, e) -> v ^ " *= " ^ string_of_sexpr e
  | SCall(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_sexpr el) ^ ")"
  | SNoexpr -> ""
                          ) ^ ")"
```

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

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

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

8.3.6. codegen.ml (Aaron Jackson)

```
module L = Llvm
module A = Ast
open Sast

module StringMap = Map.Make(String)

(* translate : Sast.program -> Llvm.module *)
let translate (globals, functions) =
  let context    = L.global_context () in
  let llmem = L.MemoryBuffer.of_file "mx.bc" in
  let llm = Llvm_bitreader.parse_bitcode context llmem in
```

```
  let the_module = L.create_module context "MX" in

  let i32_t      = L.i32_type    context
  and i8_t       = L.i8_type     context
  and i1_t       = L.i1_type     context
  and float_t    = L.double_type context
  and string_t   = L.pointer_type (L.i8_type context)
  and void_t     = L.void_type   context
  and matrix_t   = L.pointer_type (match L.type_by_name llm "struct.Matrix"
with
      None -> raise (Failure "struct.Matrix not defined")
    | Some t -> t)
  in


  (* Return the LLVM type for a MX type *)
  let ltype_of_typ = function
      A.Int   -> i32_t
    | A.Bool  -> i1_t
    | A.String-> string_t
    | A.Float -> float_t
    | A.Void  -> void_t
    | A.Matrix _ -> matrix_t
  in

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

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

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

```
let init_matrix_t =
    L.function_type matrix_t [| i32_t; i32_t |] in
let init_matrix_f =
    L.declare_function "initMatrix" init_matrix_t the_module in

let store_matrix_t =
    L.function_type matrix_t [|matrix_t; i32_t|] in
let store_matrix_f =
    L.declare_function "store" store_matrix_t the_module in

let print_matrix_t =
    L.function_type matrix_t [|matrix_t|] in
let print_matrix_f =
    L.declare_function "display" print_matrix_t the_module in

let transpose_t =
    L.function_type matrix_t [|matrix_t|] in
let transpose_f =
    L.declare_function "transpose" transpose_t the_module in

let mxAdd_t =
    L.function_type matrix_t [|matrix_t; matrix_t|] in
let mxAdd_f =
    L.declare_function "mxAdd" mxAdd_t the_module in

let mxSub_t =
    L.function_type matrix_t [|matrix_t; matrix_t|] in
let mxSub_f =
    L.declare_function "mxSub" mxSub_t the_module in

let mxMult_t =
    L.function_type matrix_t [|matrix_t; matrix_t|] in
let mxMult_f =
    L.declare_function "mxMult" mxMult_t the_module in

let identity_t =
    L.function_type matrix_t [|i32_t|] in
let identity_f =
    L.declare_function "identity" identity_t the_module in

let mxScale_t =
    L.function_type matrix_t [|matrix_t;i32_t|] in
```

```
  let mxScale_f =
      L.declare_function "mxScale" mxScale_t the_module in

  (* Dummy Function to test that built in functions can take multiple
arguments *)
  let twoFunc_t =
      L.function_type matrix_t [|i32_t;i32_t|] in
  let twoFunc_f =
      L.declare_function "twoFunc" twoFunc_t the_module in

  let transformation_t =
      L.function_type matrix_t [|matrix_t;i32_t|] in
  let transformation_f =
      L.declare_function "transformation" transformation_t the_module in

  let numCols_t =
      L.function_type i32_t [|matrix_t|] in
  let numCols_f =
      L.declare_function "numCols" numCols_t the_module in

  let numRows_t =
      L.function_type i32_t [|matrix_t|] in
  let numRows_f =
      L.declare_function "numRows" numRows_t the_module in

  let pi_t =
      L.function_type float_t [||] in
  let pi_f =
      L.declare_function "pi" pi_t the_module in


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

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

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

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

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

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

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


    (* Construct code for an expression; return its value *)
    let rec expr builder ((_, e) : sexpr) = match e with
```

```
            SLiteral i                    -> L.const_int i32_t i
      | SBoolLit b                  -> L.const_int i1_t (if b then 1 else
0)
      | SStringlit l                -> L.build_global_stringptr l "tmp"
builder
      | SFliteral l                 -> L.const_float_of_string float_t l
      | SNoexpr                     -> L.const_int i32_t 0
      | SId s                       -> L.build_load (lookup s) s builder
      | SMx (l, rows, cols)         ->
                                        let m = L.build_call init_matrix_f
[| L.const_int i32_t rows; L.const_int i32_t cols |] "init_matrix" builder
in
                                        let flat_list = List.flatten l in
                                        ignore( List.map (fun v ->
L.build_call store_matrix_f [| m ; L.const_int i32_t v |] "store_matrix"
builder) flat_list ); m

      | SAssign (s, e) -> let e' = expr builder e in
                          ignore(L.build_store e' (lookup s) builder); e'

                         (* let e1' = expr builder e in
                          let e2' = L.build_load (lookup s) s builder in
                          let
                          e' = L.build_add e1' e2' "tmp" builder in
                          ignore(L.build_store e' (lookup s) builder); e'
*)

      (* Assignment Operators below *)

      | SPlusassign (s, e) ->
                              let e1' = (expr builder e) in
                              let e2' = L.build_load (lookup s) s builder
in
                              let
                              e' = L.build_add e1' e2' "tmp" builder in
                              ignore(L.build_store e' (lookup s) builder);
e'

      | SMinusassign (s, e) ->
                              let e1' = (expr builder e) in
                              let e2' = L.build_load (lookup s) s builder
in
                              let
```

```
                              e' = L.build_sub e2' e1' "tmp" builder in
                              ignore(L.build_store e' (lookup s) builder);
e'

      | STimesassign (s, e) ->
                              let e1' = (expr builder e) in
                              let e2' = L.build_load (lookup s) s builder
in
                              let
                              e' = L.build_mul e1' e2' "tmp" builder in
                              ignore(L.build_store e' (lookup s) builder);
e'

    (* All cases of Float op Int where Flaot is lhs and Int is rhs and Cast
the int in question to a float *)

    | SBinop (((A.Float,_ ) as e1), op, ((A.Int,_) as e2)) ->
        let e1' = expr builder e1
        and e2' = expr builder e2 in
        (match op with
          A.Add     -> L.build_fadd
        | A.Sub     -> L.build_fsub
        | A.Mult    -> L.build_fmul
        | A.Div     -> L.build_fdiv
        | A.Equal   -> L.build_fcmp L.Fcmp.Oeq
        | A.Neq     -> L.build_fcmp L.Fcmp.One
        | A.Less    -> L.build_fcmp L.Fcmp.Olt
        | A.Leq     -> L.build_fcmp L.Fcmp.Ole
        | A.Greater -> L.build_fcmp L.Fcmp.Ogt
        | A.Geq     -> L.build_fcmp L.Fcmp.Oge
        | A.And | A.Or | A.Mxadd | A.Mxsub | A.Mxtimes | A.Mxscale ->
            raise (Failure "internal error: semant should have rejected
and/or on float")
        ) e1' (L.build_uitofp e2' float_t "tmp" builder) "tmp" builder

  (* Same as above but for instance of Int op Float where Int is lhs and
Float is rhs *)

    | SBinop (((A.Int,_ ) as e1), op, ((A.Float,_) as e2)) ->
        let e1' = expr builder e1
        and e2' = expr builder e2 in
        (match op with
          A.Add     -> L.build_fadd
```

```
        | A.Sub     -> L.build_fsub
        | A.Mult    -> L.build_fmul
        | A.Div     -> L.build_fdiv
        | A.Equal   -> L.build_fcmp L.Fcmp.Oeq
        | A.Neq     -> L.build_fcmp L.Fcmp.One
        | A.Less    -> L.build_fcmp L.Fcmp.Olt
        | A.Leq     -> L.build_fcmp L.Fcmp.Ole
        | A.Greater -> L.build_fcmp L.Fcmp.Ogt
        | A.Geq     -> L.build_fcmp L.Fcmp.Oge
        | A.And | A.Or | A.Mxadd | A.Mxsub | A.Mxtimes | A.Mxscale ->
            raise (Failure "internal error: semant should have rejected
and/or on float")
        ) (L.build_uitofp e1' float_t "tmp" builder) e2' "tmp" builder

    | SBinop (((A.Float,_ ) as e1), op, ((A.Float,_) as e2)) ->
        let e1' = expr builder e1
        and e2' = expr builder e2 in
        (match op with
          A.Add     -> L.build_fadd
        | A.Sub     -> L.build_fsub
        | A.Mult    -> L.build_fmul
        | A.Div     -> L.build_fdiv
        | A.Equal   -> L.build_fcmp L.Fcmp.Oeq
        | A.Neq     -> L.build_fcmp L.Fcmp.One
        | A.Less    -> L.build_fcmp L.Fcmp.Olt
        | A.Leq     -> L.build_fcmp L.Fcmp.Ole
        | A.Greater -> L.build_fcmp L.Fcmp.Ogt
        | A.Geq     -> L.build_fcmp L.Fcmp.Oge
        | A.And | A.Or | A.Mxadd | A.Mxsub | A.Mxtimes | A.Mxscale ->
            raise (Failure "internal error: semant should have rejected
and/or on float")
        ) e1' e2' "tmp" builder

      | SBinop (e1, op, e2) ->
        let e1' = expr builder e1
        and e2' = expr builder e2 in
        (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.And     -> L.build_and e1' e2' "tmp" builder
        | A.Or      -> L.build_or 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.Mxadd   -> L.build_call mxAdd_f [| e1'; e2' |] "mxAdd" builder
    | A.Mxsub   -> L.build_call mxSub_f [| e1'; e2' |] "mxSub" builder
    | A.Mxtimes -> L.build_call mxMult_f [| e1'; e2' |] "mxMult" builder
    | A.Mxscale -> L.build_call mxScale_f [| e1'; e2' |] "mxScale" builder

    )


      | SUnop(op, ((t, _) as e)) ->
          let e' = expr builder e in
        (match op with
          A.Neg when t = A.Float -> L.build_fneg e' "tmp" builder
        | A.Neg                  -> L.build_neg e' "tmp" builder
    | A.Transpose            -> L.build_call transpose_f [| e' |]
"transpose" builder
    | A.Not                  -> L.build_not e' "tmp" builder)

      | SCall ("print", [e]) | SCall ("printb", [e]) ->
        L.build_call printf_func [| int_format_str ; (expr builder e) |]
          "printf" builder

      | SCall ("printbig", [e]) ->
        L.build_call printbig_func [| (expr builder e) |] "printbig"
builder

      | SCall ("print_matrix", [e]) ->
        L.build_call print_matrix_f [| (expr builder e) |] "printbig"
builder

      | SCall ("transpose", [e]) ->
        L.build_call transpose_f [| (expr builder e) |] "transpose" builder

      | SCall ("printf", [e]) ->
        L.build_call printf_func [| float_format_str ; (expr builder e) |]
          "printf" builder

      | SCall ("prints", [e]) ->
```

```
      L.build_call printf_func [| str_format_str ; (expr builder e) |]
"printf" builder

      | SCall ("twoFunc", [e1; e2]) ->
      L.build_call twoFunc_f [| (expr builder e1); (expr builder e2) |]
"twoFunc" builder

      | SCall ("transformation", [e1; e2]) ->
      L.build_call transformation_f [| (expr builder e1); (expr builder e2)
|] "transformation" builder

      | SCall ("identity", [e]) ->
      L.build_call identity_f [| (expr builder e) |] "identity" builder

      | SCall ("numCols", [e]) ->
      L.build_call numCols_f [| (expr builder e) |] "numCols" builder

      | SCall ("numRows", [e]) ->
      L.build_call numRows_f [| (expr builder e) |] "numRows" builder

      | SCall ("pi", _) ->
      L.build_call pi_f [|||] "pi" builder

      | SCall (f, args) ->
         let (fdef, fdecl) = StringMap.find f function_decls in
       let llargs = List.rev (List.map (expr builder) (List.rev args)) in
       let result = (match fdecl.styp with
                       A.Void -> ""
                     | _ -> f ^ "_result") in
         L.build_call fdef (Array.of_list llargs) result builder
    in

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

    (* Build the code for the given statement; return the builder for
```

```
     the statement's successor (i.e., the next instruction will be built
     after the one generated by this call) *)

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

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

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

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

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

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

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

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

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

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

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

  List.iter build_function_body functions;
  the_module
```

8.3.7. mx.c (Wilderness Oberman, Aaron Jackson)

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>

int debug = 0;

static void die(const char *message) {
    perror(message);
    exit(1);
}

typedef struct Matrix {
  int num_rows;
  int num_cols;
  int *matrixAddr;
  int buildPosition;
} Matrix;

Matrix *store(Matrix *m, int value) {
```

```
  int position = m->buildPosition;
  int curr_row = position / m->num_cols;
  int curr_col = position % m->num_cols;

  m->matrixAddr[position] = value;
  m->buildPosition = m->buildPosition + 1;
  return m;
}

Matrix *initMatrix( int rows, int cols ) {
  int size = rows * cols;
  int *p = malloc(sizeof(int*)*size);

  for (int i = 0; i <= size; i++) {
    p[i] = 0;
  }

  Matrix *result = malloc(sizeof(struct Matrix));
  result->num_cols = cols;
  result->num_rows = rows;
  result->matrixAddr = p;
  result->buildPosition = 0;

  return result;
}

int get(struct Matrix *m, int r, int c) {
  //get m[r][c]
  int kill = 0;
  if (r > ((m->num_rows) - 1)) {
    perror("row index out of range when setting matrix ");
    kill = 1;
  }
  if (c > ((m->num_cols) - 1)) {
    perror("col index out of range when setting matrix ");
    kill = 1;
  }
  if(kill == 1) {
    die("");
```

```
  }
  int idx = c + (r * (m->num_cols));
  return m->matrixAddr[idx];
}

void set(struct Matrix *m, int r, int c, double v) {
  //set m[r][c] to v
  int kill = 0;
  if (r > ((m->num_rows) - 1)) {
    perror("row index out of range when setting matrix ");
    kill = 1;
  }
  if (c > ((m->num_cols) - 1)) {
    perror("col index out of range when setting matrix ");
    kill = 1;
  }
  if(kill == 1) {
    die("");
  }
  int idx = c + (r * (m->num_cols));
  m->matrixAddr[idx] = v;
}

int numCols(Matrix *input) {
  int cols = input->num_cols;
  return cols;
}

int numRows(Matrix *input) {
  int rows = input->num_rows;
  return rows;
}

double pi(int a) {
  double pi = 2*acos(0.0);
  return pi;
}

Matrix *mxAdd(Matrix *lhs, Matrix *rhs) {
```

```
  //check dimensions
  if (lhs->num_rows != rhs->num_rows || lhs->num_cols != rhs->num_cols) {
    die("Addition size mismatch.");
  }
  int rows = lhs->num_rows;
  int cols= lhs->num_cols;
  Matrix *result = initMatrix(rows, cols);
  for(int i = 0; i < rows; i++) {
    for(int j = 0; j < cols; j++) {
        int sum = get(lhs, i, j) + get(rhs, i, j);
        set(result, i, j, sum);
    }
  }
  return result;
}

Matrix *mxSub(Matrix *lhs, Matrix *rhs) {
  //check dimensions
  if (lhs->num_rows != rhs->num_rows || lhs->num_cols != rhs->num_cols) {
    die("Subtraction size mismatch.");
  }
  int rows = lhs->num_rows;
  int cols = lhs->num_cols;
  Matrix *result = initMatrix(rows, cols);
  for(int i = 0; i < rows; i++) {
    for(int j = 0; j < cols; j++) {
        int res = get(lhs, i, j) - get(rhs, i, j);
        set(result, i, j, res);
    }
  }
  return result;
}

Matrix *mxMult(Matrix *lhs, Matrix *rhs) {
  //check dimensions
  if (lhs->num_cols != rhs->num_rows) {
    die("Multiplication size mismatch.");
  }
```

```
  int rows = lhs->num_rows;
  int cols = rhs->num_cols;
  Matrix *result = initMatrix(rows, cols);
  for(int i = 0; i < rows; i++) {
    for(int j = 0 ; j < cols; j++) {
      for(int k = 0; k < rhs->num_rows; k++) {
        set(result, i, j, get(result, i, j) + (get(lhs, i, k) * get(rhs, k, j)));
      }
    }
  }
  return result;
}

Matrix *mxScale(Matrix *input, int scalar) {
  int rows = input->num_rows;
  int cols = input->num_cols;
  Matrix *result = initMatrix(rows, cols);
  for(int i = 0; i < rows; i++) {
    for(int j = 0 ; j < cols; j++) {
      int product = scalar * get(input,i,j);
      set(result, i, j, product);
    }
  }
  return result;
}

Matrix *identity(int dim) {
  //create an NxN identity matrix
  int rows = dim;
  int cols = dim;
  Matrix *result = initMatrix(rows, cols);
  for(int i = 0; i < rows; i++) {
    for(int j = 0 ; j < cols; j++) {
      if(i == j) {
        set(result, i, j, 1);
      } else {
        set(result, i, j, 0);
      }
    }
```

```
  }
  return result;
}

Matrix *transpose(Matrix *input) {
  //switch rows and cols, get empty(i.e., zeroed matrix of transposed size, then
fill)
  int rows = input->num_rows;
  int cols = input->num_cols;

  Matrix *result = initMatrix(cols, rows);
  for(int i = 0; i < rows; i++) {
    for(int j = 0; j < cols; j++) {
        set(result, j, i, get(input,i,j));
    }
  }
  return result;
}

Matrix *transformation(Matrix *input, int num) {
  //check dimensions
  if(input->num_rows != 2) {
    die("Invalid matrix size for 2D transformations");
  }
  int rows = input->num_rows;
  int cols = input->num_cols;
  Matrix *result = initMatrix(rows, cols);
  Matrix *tmp = initMatrix(2, 2);
  switch(num) {
    case 1 :
      //Reflection in line y = x
      set(tmp, 0, 0, 0);
      set(tmp, 0, 1, 1);
      set(tmp, 1, 0, 1);
      set(tmp, 1, 1, 0);
      result = mxMult(tmp, input);
      break;
    case 2 :
      //Reflection in line y = -x
```

```
    set(tmp, 0, 0, 0);
    set(tmp, 0, 1, -1);
    set(tmp, 1, 0, -1);
    set(tmp, 1, 1, 0);
    result = mxMult(tmp, input);
    break;
  case 3 :
    //Reflection in x-axis
    set(tmp, 0, 0, 1);
    set(tmp, 0, 1, 0);
    set(tmp, 1, 0, 0);
    set(tmp, 1, 1, -1);
    result = mxMult(tmp, input);
    break;
  case 4 :
    //Reflection in y-axis
    set(tmp, 0, 0, -1);
    set(tmp, 0, 1, 0);
    set(tmp, 1, 0, 0);
    set(tmp, 1, 1, 1);
    result = mxMult(tmp, input);
    break;
  case 5 :
    //Rotation 90° clockwise
    set(tmp, 0, 0, 0);
    set(tmp, 0, 1, 1);
    set(tmp, 1, 0, -1);
    set(tmp, 1, 1, 0);
    result = mxMult(tmp, input);
    break;
  case 6 :
    //Rotation 180°
    set(tmp, 0, 0, -1);
    set(tmp, 0, 1, 0);
    set(tmp, 1, 0, 0);
    set(tmp, 1, 1, -1);
    result = mxMult(tmp, input);
    break;
  case 7 :
```

```
      //Rotation 90° anticlockwise
      set(tmp, 0, 0, 0);
      set(tmp, 0, 1, -1);
      set(tmp, 1, 0, 1);
      set(tmp, 1, 1, 0);
      result = mxMult(tmp, input);
      break;
    default :
      printf("Invalid input.");
  }
  return result;
}

void display(Matrix *input) {
  int rows = input->num_rows;
  int cols = input->num_cols;
  printf("\n");
  for(int i = 0; i < rows; i++) {
    for(int j = 0; j < cols; j++) {
      if (j == 0) {
        printf("[ %i,", get(input, i, j));
      } else if (j == cols - 1) {
        printf(" %i ]", get(input, i, j));
      } else {
        printf(" %i,", get(input, i, j));
      }
    }
    printf("\n");
  }
}

void twoFunc(int a, int b){
  printf("%d\n%d\n",a,b);
}
```

8.3.8. Mx.ml (Aaron Jackson)

```
(* Top-level of the MX compiler: scan & parse the input,
   check the resulting AST and generate an SAST from it, generate LLVM IR,
   and dump the module *)

type action = Ast | Sast | LLVM_IR | Compile

let () =
  let action = ref Compile in
  let set_action a () = action := a in
  let speclist = [
    ("-a", Arg.Unit (set_action Ast), "Print the AST");
    ("-s", Arg.Unit (set_action Sast), "Print the SAST");
    ("-l", Arg.Unit (set_action LLVM_IR), "Print the generated LLVM IR");
    ("-c", Arg.Unit (set_action Compile),
      "Check and print the generated LLVM IR (default)");
  ] in
  let usage_msg = "usage: ./mx.native [-a|-s|-l|-c] [file.mx]" in
  let channel = ref stdin in
  Arg.parse speclist (fun filename -> channel := open_in filename)
usage_msg;

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

8.3.9. Makefile (Rashel Rojas, Wilderness Oberman, Aaron Jackson, Mauricio Guerrero)

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

.PHONY: test
test: all testall.sh
```

```
    ./testall.sh

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

.PHONY : all
all : mx.native mx.o

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

mx.native : mx.bc
    opam exec -- \
    ocamlbuild -use-ocamlfind mx.native -pkgs llvm,llvm.analysis,llvm.bitreader

# "make clean" removes all generated files

mx : mx.c
    cc -o mx -DBUILD_TEST mx.c

mx.bc : mx.c
    clang -emit-llvm -o mx.bc -c mx.c -Wno-varargs

.PHONY : clean
clean :
    ocamlbuild -clean
    rm -rf testall.log ocamlllvm *.diff *.exe *.s *.ll *.bc *.o tests/*.exe
tests/*.s tests/*.ll
```

8.4. Tests

8.4.1. ./testall.sh

```
#!/bin/sh

# Authored by Mauricio Guerrero, Aaron Jackson
```

```
# Citation: based on MicroC test script.

# Regression testing script for MX
# Step through a list of files
#  Compile, run, and check the output of each expected-to-work test
#  Compile and check the error of each expected-to-fail test
# Path to the LLVM interpreter
LLI="lli"
#LLI="/usr/local/opt/llvm/bin/lli"
# Path to the LLVM compiler
LLC="llc"
# Path to the C compiler
CC="cc"
# Path to the mx compiler.  Usually "./mx.native"
# Try "_build/mx.native" if ocamlbuild was unable to create a symbolic
link.
MX="./mx.native"
#MX="_build/mx.native"
# Set time limit for all operations
ulimit -t 30
globallog=testall.log
rm -f $globallog
error=0
globalerror=0
keep=0
Usage() {
   echo "Usage: testall.sh [options] [.mx files]"
   echo "-k    Keep intermediate files"
   echo "-h    Print this help"
   exit 1
}
SignalError() {
   if [ $error -eq 0 ] ; then
   echo "FAILED"
   error=1
   fi
   echo "  $1"
}
# Compare <outfile> <reffile> <difffile>
```

```
# Compares the outfile with reffile.  Differences, if any, written to
difffile
Compare() {
   generatedfiles="$generatedfiles $3"
   echo diff -b $1 $2 ">" $3 1>&2
   diff -b "$1" "$2" > "$3" 2>&1 || {
   SignalError "$1 differs"
   echo "FAILED $1 differs from $2" 1>&2
   }
}
# Run <args>
# Report the command, run it, and report any errors
Run() {
   echo $* 1>&2
   eval $* || {
   SignalError "$1 failed on $*"
   return 1
   }
}
# RunFail <args>
# Report the command, run it, and expect an error
RunFail() {
   echo $* 1>&2
   eval $* && {
   SignalError "failed: $* did not report an error"
   return 1
   }
   return 0
}
Check() {
   error=0
   basename=`echo $1 | sed 's/.*\\///
                           s/.mx//'`
   reffile=`echo $1 | sed 's/.mx$//'`
   basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."
   echo -n "$basename..."
   echo 1>&2
   echo "###### Testing $basename" 1>&2
   generatedfiles=""
```

```
   generatedfiles="$generatedfiles ${basename}.ll ${basename}.s
${basename}.exe ${basename}.out" &&
   Run "$MX" "$1" ">" "${basename}.ll" &&
   Run "$LLC" "-relocation-model=pic" "${basename}.ll" ">" "${basename}.s"
&&
   Run "$CC" "-o" "${basename}.exe" "${basename}.s" "mx.c" "-lm" &&
   Run "./${basename}.exe" > "${basename}.out" &&
   Compare ${basename}.out ${reffile}.out ${basename}.diff
   # Report the status and clean up the generated files
   if [ $error -eq 0 ] ; then
   if [ $keep -eq 0 ] ; then
       rm -f $generatedfiles
   fi
   echo "OK"
   echo "###### SUCCESS" 1>&2
   else
   echo "###### FAILED" 1>&2
   globalerror=$error
   fi
}
CheckFail() {
   error=0
   basename=`echo $1 | sed 's/.*\\///
                           s/.mx//'`
   reffile=`echo $1 | sed 's/.mx$//'`
   basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."
   echo -n "$basename..."
   echo 1>&2
   echo "###### Testing $basename" 1>&2
   generatedfiles=""
   generatedfiles="$generatedfiles ${basename}.err ${basename}.diff" &&
   RunFail "$MX" "<" $1 "2>" "${basename}.err" ">>" $globallog &&
   Compare ${basename}.err ${reffile}.err ${basename}.diff
   # Report the status and clean up the generated files
   if [ $error -eq 0 ] ; then
   if [ $keep -eq 0 ] ; then
       rm -f $generatedfiles
   fi
   echo "OK"
   echo "###### SUCCESS" 1>&2
```

```
   else
   echo "###### FAILED" 1>&2
   globalerror=$error
   fi
}
while getopts kdpsh c; do
   case $c in
   k) # Keep intermediate files
       keep=1
       ;;
   h) # Help
       Usage
       ;;
   esac
done
shift `expr $OPTIND - 1`
LLIFail() {
 echo "Could not find the LLVM interpreter \"$LLI\"."
 echo "Check your LLVM installation and/or modify the LLI variable in
testall.sh"
 exit 1
}
which "$LLI" >> $globallog || LLIFail
# if [ ! -f printbig.o ]
# then
#     echo "Could not find printbig.o"
#     echo "Try \"make printbig.o\""
#     exit 1
# fi
if [ $# -ge 1 ]
then
   files=$@
else
   files="tests/test-*.mx tests/fail-*.mx"
fi
for file in $files
do
   case $file in
   *test-*)
       Check $file 2>> $globallog
```

```
        ;;
    *fail-*)
        CheckFail $file 2>> $globallog
        ;;
    *)
        echo "unknown file type $file"
        globalerror=1
        ;;
    esac
done
exit $globalerror
```

8.4.2. fail-add1.mx

```
int main() {
    Matrix m1;
    Matrix m2;
    Matrix m3;

    m1 = [[1,3],[7,1,3]];
    m2 = [[5,4],[3,6]];

    m3 = m1 +. m2;
}
```

Expected output:
fail-add1.err

```
Fatal error: exception Failure("Matrix rows are not all the
same length")
```

8.4.3. fail-add2.mx

```
int main() {
    Matrix m;
    String s;
    Matrix sum;
```

```
    m = [[6,7,8],[9,4,5],[5,4,4]];
    s = "pineapples";

    sum = m +. s; /* Fail: add a matrix and string */
}
```

Expected output:
fail-add2.err

```
Fatal error: exception Failure("illegal binary operator
matrix +. string in m +. s")
```

8.4.4. fail-add3.mx

```
int main() {
    Matrix m;
    int x;
    Matrix sum;

    m = [[9,4],[1,5]];
    x = 45;

    sum = m +. x; /* Fail: add a matrix and int */
}
```

Expected output:
fail-add3.err

```
Fatal error: exception Failure("illegal binary operator
matrix +. int in m +. x")
```

8.4.5. fail-add4.mx

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

    print(i +. j);
}
```

Expected output:

fail-add4.err

```
Fatal error: exception Failure("illegal binary operator int
+. float in i +. j")
```

8.4.6. fail-arithmetic2.mx

```
int main() {
    String s;
    int i;
    int j;

    i = 50;
    j = 3 + i / s;
    print(j);
}
```

Expected output:
fail-arithmetic2.err

```
Fatal error: exception Failure("illegal binary operator int /
string in i / s")
```

8.4.7. fail-arithmetic3.mx

```
int main() {
    String s;
    int i;
    bool b;
    int j;

    j = b - i;
    print(j);
}
```

Expected Output:
fail-arithmetic3.err

```
Fatal error: exception Failure("illegal binary operator bool
- int in b - i")
```

8.4.8. fail-arithmetic.mx

```
int main() {
    int x;
    x = 10 + 3 -;
}
```

Expected Output:
fail-arithmetic.err

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

8.4.9. fail-assign1.mx

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

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

Expected Output:
fail-assign1.err

```
Fatal error: exception Failure("illegal assignment int = bool
in i = false")
```

8.4.10. fail-assign2.mx

```
int main()
{
    Matrix m;
    int num;

    m = [[1,3],[10,20]];
    num = 14;
    m = 20; /* Fail: assigning a matrix to an integer */
    num = 5;
}
```

Expected Output:
fail-assign2.err

```
Fatal error: exception Failure("illegal assignment matrix =
int in m = 20")
```

8.4.11. fail-assign3.mx

```
int main()
{
    String s;

    s = "hello";
    s = [[1,2,3],[6,4,5],[9,3,5]];
}
```

Expected Output:
fail-assign3.err

```
Fatal error: exception Failure("illegal assignment string =
matrix in s =
[1,2,3,6,4,5,9,3,5]
")
```

8.4.12. fail-assignop.mx

```
int main() {
    String s;
    int i;

    s = "hello";
    i = 1;
    i += s;
}
```

Expected Output:
fail-assignop.err

```
Fatal error: exception Failure("illegal assignment int =
string in i += s")
```

8.4.13. fail-dead.mx

```
int main() {
    int i;
    return 1;
    i = 1;
}
```

Expected Output:
fail-dead.err

```
Fatal error: exception Failure("nothing may follow a return")
```

8.4.14. fail-for.mx

```
int main() {
    int i;
    for (i = 0 ; k < 10 ; i = i + 1) {
        if (i > 5) {
            print(i * 2);
        }
    }

}
```

Expected Output:
fail-for.err

```
Fatal error: exception Failure("undeclared identifier k")
```

8.4.15. fail-funcarg1.mx

```
int foo(int x, int y, int z) {
    int sum;

    sum = x + y + z;
    return sum;
}

int main() {
    int secret;
```

```
    secret = 100;

    if (secret > 100) {
        print(secret);
    } else {
        print(foo(3, 5));
    }
}
```

Expected Output:
fail-funcarg1.err

```
Fatal error: exception Failure("expecting 3 arguments in
foo(3, 5)")
```

8.4.16. fail-funcarg2.mx

```
bool foo(int i, float f, bool tf) {
    if (i > 0 && f > 3.5) {
        return tf;
    } else {
        return false;
    }
}


int main() {
    bool result;
    bool b;

    b = true;
    result = foo(5, b, 10.0);
    printb(result);
}
```

Expected Output:
fail-funcarg2.err

```
Fatal error: exception Failure("illegal argument found bool
expected float in b")
```

8.4.17. fail-funcarg3.mx

```
int foo(int a, int b, int c, int b) {
    return a * b / a;
}

int main() {
    int x;

    x = foo(3, 4, 5, 6);
}
```

Expected Output:
fail-funcarg3.err

```
Fatal error: exception Failure("duplicate formal b")
```

8.4.18. fail-identity1.mx

```
int main() {
    Matrix m;
    String num;

    num = "3";
    m = identity(num);
}
```

Expected Output:
fail-identity1.err

```
Fatal error: exception Failure("illegal argument found string
expected int in num")
```

8.4.19. fail-identity2.mx

```
int main() {
    Matrix m;
    float f;

    f = 3.42;
    m = identity(f);
}
```

Expected Output:
fail-identity2.err

```
Fatal error: exception Failure("illegal argument found float
expected int in f")
```

8.4.20. fail-if.mx

```
int main() {
    int x;

    x = 5;
    if ("true") {
        print(x);
    }
}
```

Expected Output:
fail-if.err

```
Fatal error: exception Failure("expected Boolean expression
in true")
```

8.4.21. fail-matrix-declaration.mx

```
int main() {
    Matrix m;
    m = [[2,2],[2,2]; /* parsing error */
}
```

Expected Output:
fail-matrix-declaration.err

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

8.4.22. fail-matrixfunction1.mx

```
int main() {
    Matrix m;
    int i;

    m = [[30,40,50,20],[1,3,2,2],[4,5,6,3]];
    i = 10;
```

```
    i.numRows(m);
}
```

Expected Output:
fail-matrixfunction1.err

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

8.4.23. fail-matrixfunction2.mx

```
int main() {
    Matrix m;
    Matrix n;

    m = [[1,2],[3,4]];
    n = [[12,17],[34,21]];

    m.transformation(n);
}
```

Expected Output:
fail-matrixfunction2.err

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

8.4.24. fail-mult1.mx

```
int main() {
    Matrix m1;
    Matrix m2;
    Matrix m3;

    m1 = [[4,3,5],[2,1]];
    m2 = [[2,8,9],[2,4,5],[7,7,4]];

    m3 = m1 *. m2;
}
```

Expected Output:
fail-mult1.err

```
Fatal error: exception Failure("Matrix rows are not all the
same length")
```

8.4.25. fail-mult2.mx

```
int main() {
    String str;
    Matrix m;
    Matrix result;

    str = "34";
    m = [[6,7,34,1],[3,4,9,12]];

    result = str *. m;
}
```

Expected output:
fail-mult2.err

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

8.4.26. fail-mult3.mx

```
int main() {
    Matrix m1;
    Matrix m2;
    Matrix m3;

    m1 = [[4,3],[2,5]];
    m2 = [[],[2]];

    m3 = m1 *. m2;
}
```

Expected output:
fail-mult3.err

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

8.4.27. fail-mult4.mx

```
int main() {
    String num;
    Matrix mat;
```

```
    Matrix multi;

    num = "34";
    mat = [[1,2],[3,4],[4,5],[5,8]];

    multi = mat *. num;
}
```

Expected output:
fail-mult4.err

```
Fatal error: exception Failure("illegal binary operator
matrix *. string in mat *. num")
```

8.4.28. fail-nomain.mx

```
String foo(String s) {
    return s;
}
```

Expected output:
fail-nomain.err

```
Fatal error: exception Failure("unrecognized function main")
```

8.4.29. fail-numcols.mx

```
int main() {
    Matrix m;
    int i;
    int c;

    m = [[30,40,50,20],[1,3,2,2],[4,5,6,3]];
    i = 15;

    c = numRows(i);
}
foo(String s) {
    return s;
}
```

Expected output:

fail-numcols.err

```
Fatal error: exception Failure("illegal argument found int
expected matrix in i")
```

8.4.30. fail-numrows.mx

```
int main() {
    Matrix m;
    int i;

    m = [[30,40,50,20],[1,3,2,2],[4,5,6,3]];
    i = 30;

    numRows(i);
}
```

Expected output:
fail-numrows.err

```
Fatal error: exception Failure("illegal argument found int
expected matrix in i")
```

8.4.31. fail-opand.mx

```
int main() {
    String s;
    int i;
    bool b;
    bool bb;
    int j;

    i = 3;
    j = 4;

    printb(i > 0 && j);
}
```

Expected output:
fail-opand.err

```
Fatal error: exception Failure("illegal binary operator bool
&& int in i > 0 && j")
```

8.4.32. fail-printb.mx

```
int main() {
    bool bb;
    String s;

    bb = true;
    s = "hello world!";
    printb(bb);
    printb(s);
}
```

Expected output:
fail-printb.err

```
Fatal error: exception Failure("illegal argument found string
expected bool in s")
```

8.4.33. fail-printmatrix1.mx

```
int main() {
    Matrix m;
    int i;

    m = [[1,3,5,6],[3,9,7,9],[4,4,4,5],[6,7,6,4],[1,2,3,4]];
    i = 3;

    print_matrix(i);
}
```

Expected output:
fail-printmatrix1.err

```
Fatal error: exception Failure("illegal argument found int
expected matrix in i")
```

8.4.34. fail-print.mx

```
int main() {
```

```
    bool bb;

    bb = false;
    print(bb);
}
```

Expected output:
fail-print.err

```
Fatal error: exception Failure("illegal argument found bool
expected int in bb")
```

8.4.35. fail-prints.mx

```
int main() {
    String s;
    int j;

    j = 10;
    s = "hello world!";
    print(j);
    prints(j);
}
```

Expected output:
fail-prints.err

```
Fatal error: exception Failure("illegal argument found int
expected string in j")
```

8.4.36. fail-reservedfuncs.mx

```
Matrix identity() {
    Matrix m;

    m = [[1,0,0],[0,1,0],[0,0,1]];
    return m;
}

int main() {
    Matrix n;

    n = identity();
```

```
    print_matrix(n);
}
```

Expected output:
fail-reservedfuncs.err

```
Fatal error: exception Failure("function identity may not be
defined")
```

8.4.37. fail-scalemult1.mx

```
int main() {
    int x;
    Matrix m;
    Matrix result;

    x = 3;
    m = [[4,5,3], [5,5,6,7,8]];

    result = m **. x;
}
```

Expected output:
fail-scalemul1t.err

```
Fatal error: exception Failure("Matrix rows are not all the
same length")
```

8.4.38. fail-scalemult2.mx

```
int main() {
    float x;
    Matrix m;
    Matrix result;

    x = 3.2;
    m = [[4,5,3],[5,5,6]];

    result = m **. x;  /* Fail: scalar multiplication between
float and matrix */
}
```

Expected output:
fail-scalemult2.err

```
Fatal error: exception Failure("Matrix rows are not all the
same length")
```

8.4.39. fail-subtract1.mx

```
int main() {
    Matrix m1;
    Matrix m2;
    Matrix m3;

    m1 = [[1,2,3,4],[4,5,2,3],[7,5,2,3]];
    m2 = [[4,3,2,1],[2,3],[5,5]];

    m3 = m1 -. m2;
}
```

Expected output:
fail-subtract1.err

```
Fatal error: exception Failure("Matrix rows are not all the
same length")
```

8.4.40. fail-subtract2.mx

```
int main() {
    Matrix m;
    int x;
    Matrix sum;

    m = [[7,88,3],[4,5,6]];
    x = 21;

    sum = m -. x; /* Fail: subtract a matrix and int */
}
```

Expected output:
fail-subtract2.err

```
Fatal error: exception Failure("illegal binary operator
```

```
matrix -. int in m -. x")
```

8.4.41. fail-transform1.mx

```
int main() {
    Matrix m;
    Matrix t;

    m = [[1,2],[3,4]];
    t = transformation(m, "1");
}
```

Expected output:
fail-transform1.err

```
Fatal error: exception Failure("illegal argument found
string expected int in 1")
```

8.4.42. Fail-transform2.mx

```
int main() {
    Matrix m;
    Matrix t;

    m = [[1,2],[3,4]];
    t = transformation(m, 4.5);
}
```

Expected output:
fail-transform2.err

```
Fatal error: exception Failure("illegal argument found float
expected int in 4.5")
```

8.4.43. fail-transpose1.mx

```
int main() {
    Matrix m;
    Matrix matrix_transposed;
    String str;

    m = [[1,2,3],[50,60,70]];
```

```
    str = "matrix";
    matrix_transposed = str';
}
```

Expected output:
fail-transpose1.err

```
Fatal error: exception Failure("illegal unary operator
'string in 'str")
```

8.4.44. fail-transpose2.mx

```
int main() {
    Matrix m;
    Matrix matrix_transposed;
    int x;

    m = [[1,2,3],[50,60,70]];
    x = 45;
    matrix_transposed = x';
}
```

Expected output:
fail-transpose2.err

```
Fatal error: exception Failure("illegal unary operator 'int
in 'x")
```

8.4.45. fail-undecvar.mx

```
int main() {
    String s;
    bool b;
    int j;

    s = "plt is a great class for all coders!";
    b = true;

    if (i > 0) {
        printb(b);
    }
```

```
}
```

Expected output:
fail-undecvar.err

```
Fatal error: exception Failure("undeclared identifier i")
```

8.4.46. fail-while1.mx

```
int main() {
    int x;

    x = 1;
    while (x) {
        print(x);
    }
}
```

Expected output:
fail-while1.err

```
Fatal error: exception Failure("expected Boolean expression
in x")
```

8.4.47. test-arithmetic2.mx

```
int sum(int x) {
    return x + 1;
}

int main() {
    int a;
    int b;

    a = 45 + 3;
    a += 5;

    print(a);
    print(a * 2);
}
```

Expected output:
test-arithmetic 2.out

```
53
106
```

8.4.48. test-arithmetic3.mx

```
int main() {
    int x;
    x = 1 + 30 - 2;
}
```

Expected output:
test-arithmetic3.out

```
No output.
```

8.4.49. test-arithmetic.mx

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

    i = 9 + -8;
    print(i);

    i = 8 * (9 / 2);
    print(i);

    i = 3 / 2 + 1;
    print(i);

    f = 4.5 + 6;  # casting
    printf(f);

    f = 3 * 7.89;
    printf(f);

    f = 11.0 / -2.75;
    printf(f);

    i = -1 / 3;
```

```
    print(i);

    i = -1  * -3;
    print(i);

    i = 1  * -3;
    print(i);

    f = -1.0 / -4.0;
    printf(f);

    f = 1.0 / -4.0;
    printf(f);

    f = 1.0 / 4.0;
    printf(f);

    f = -1.0 + -4.0;
    printf(f);
}
```

Expected output:
test-arithmetic.out

```
1
32
2
10.5
23.67
-4
0
3
-3
0.25
-0.25
0.25
-5
```

8.4.50. test-assignment2.mx

```
int main() {
    int x;
    int j;
    String s;
```

```
    x = 0;
    j = 5;
    x += j;
    j = x + 3;
    s = "notebook";
}
```

Expected output:
test-assignment2.out

```
No output.
```

8.4.51. test-assignment.mx

```
int main() {
    int i;
    int j;
    int z;
    String s;
    bool b;
    float f;
    Matrix m;

    i = 3;
    j = z = 3;
    s = "this is a string";
    b = true;
    f = 3.5794;
    m = [[1,0],[0,1]];
}
```

Expected output:
test-assignment.out

```
No output.
```

8.4.52. test-assignop.mx

```
int main() {
    String s;
    int i;
```

```
    int j;

    for (i = 0; i < 5; i += 1) {
        j += 1;
    }
    print(j);
}
```

Expected output:
test-assignop.out

```
5
```

8.4.53. test-fibonacci.mx

```
int main() {
    String s;
    int i;
    int j;

    for (i = 0; i < 5; i += 1) {
        j += 1;
    }
    print(j);
}
```

Expected output:
test-fibonacci.out

```
1
1
2
3
5
8
13
21
34
55
```

8.4.54. test-for.mx

```
int main() {
```

```
    int i;
    prints("============================================");
    prints("for(i = 0; i < 10; i = i + 1)");
    for(i = 0; i < 10; i = i + 1) {
        if(i == 5) {
            prints("i = 5");
        } else {
            print(i);
        }
    }
    prints("============================================");
    prints("for( i = 5; i > 0; i = i -1)");
    for( i = 5; i > 0; i = i -1) {
        print(i);
    }
    prints("============================================");
}
```

Expected output:
test-for.out

```
============================================
for(i = 0; i < 10; i = i + 1)
0
1
2
3
4
i = 5
6
7
8
9
============================================
for( i = 5; i > 0; i = i -1)
5
4
3
2
1
============================================
```

8.4.55. test-func.mx

```
bool foo(bool b1, bool b2) {
    return b1 && b2;
```

```
}

float multi(int a, int b, float f) {
    float m;
    m = a * b * f;
    return m;
}

bool logic(int a, int b, float f) {
    float x;

    x = multi(a, b, f);

    if (x < 0.0) {
        return true;
    } else {
        return false;
    }
}

int main() {
    bool b;
    bool bb;
    bool result;

    b = true;
    bb = false;
    result = logic(1,2,-3.0);

    printb(foo(b, bb));

    if (result == true) {
        printb(result);
    } else {
        prints("oh no!");
    }
}
```

Expected output:
test-func.out

```
0
1
```

8.4.56. test-gcd.mx

```
int gcd(int a, int b) {
      while (a != b) {
            if (a > b) a = a - b;
            else b = b - a;
      }
      return a;
}
int main()
{
      print(gcd(2,14));
      print(gcd(3,15));
      print(gcd(99,121));
      return 0;
}
```

Expected output:
test-gcd.out

```
2
3
11
```

8.4.57. test-hello.mx

```
int main(){
    String e;
    e = "Hello!";
    prints(e);
}
```

Expected output:
test-hello.out

```
Hello!
```

8.4.58. test-if.mx

```
int main(){
    int a;
    int b;
    float c;
    float d;
    a = 5;
```

```
    b = 10;
    c = 9.81;
    d = 4.9;

    prints("=============================================");
    prints("a is an integer with the value of: ");
    print(a);
    prints("=============================================");
    prints("b is an integer with the value of: ");
    print(b);
    prints("=============================================");
    prints("c is a float with the value of: ");
    printf(c);
    prints("=============================================");
    prints("d is an float with the value of: ");
    printf(d);

    if(a <= b) {

prints("=============================================");
        prints("a is less than or equal to b.");

prints("=============================================");

    }

    if(b > a) {
        prints("b is greater than a.");

prints("=============================================");
    }

    if(c > d) {
        prints("c is greater than d.");

prints("=============================================");
    }

    if(c >= d) {
        prints("c is greater than  or equal to d.");

prints("=============================================");
    }

    if(3 == 3) {
        prints("3 does equal to 3");

prints("=============================================");
    }
```

```
    if(true && true) {
        prints("true");

prints("================================================");
    }

    if(true || false) {
        prints("true || false is true ");

prints("================================================");
    }

    if(true) {
        prints("true is true");

prints("================================================");
    }

    if( 6 != 6 ) {
       prints("This was a mistake, we should fix tihs!");
    } else {
        prints("This is the right output!");

prints("================================================");
    }

    if(false) {
        prints("Huh? What happened?");
    } else {
        prints("if(false) took me to the else, which is
here!");

prints("================================================");
    }

    if( (true && true) && (6>5)) {
        prints("(true && true) && (6>5) is true");

prints("================================================");
    }


    if(false) {
        prints("(true && true) && (6>5) is true");

prints("================================================");
    } else {
        if( (true || false) || (false && false)) {
```

```
            prints("if(false) {} else {if( (true || false) ||
(false && false))}");

prints("============================================");
        }
    }


}
```

Expected output:
test-if.out

```
============================================
a is an integer with the value of:
5
============================================
b is an integer with the value of:
10
============================================
c is a float with the value of:
9.81
============================================
d is an float with the value of:
4.9
============================================
a is less than or equal to b.
============================================
b is greater than a.
============================================
c is greater than d.
============================================
c is greater than  or equal to d.
============================================
3 does equal to 3
============================================
true
============================================
true || false is true
============================================
true is true
============================================
This is the right output!
============================================
if(false) took me to the else, which is here!
============================================
(true && true) && (6>5) is true
============================================
```

```
if(false) {} else {if( (true || false) || (false && false))}
==========================================
```

8.4.59. test-logicalops.mx

```
int main()
{
    printb(true && true);
    printb(false && false);
    printb(true && false);
    printb(false && true);

    prints(">>>>>>>>>>");
    printb(true || true);
    printb(false || false);
    printb(true || false);
    printb(false || true);

    prints(">>>>>>>>>>");
    printb(!true);
    printb(!false);

    prints(">>>>>>>>>>");
    printb(true);
    printb(false);
}
```

Expected output:
test-logicalops.out

```
1
0
0
0
>>>>>>>>>>
1
0
1
1
>>>>>>>>>>
0
1
>>>>>>>>>>
1
0
```

8.4.60. test-matrixAdd.mx

```
int main(){
    Matrix n;
    Matrix b;
    Matrix a;
    n = [[1,2,4],[3,4,5]];
    a = [[1,2,4],[3,4,5]];
    b = a +. n;
    prints("=============================================");
    prints("n = ");
    print_matrix(n);
    prints("=============================================");
    prints("a = ");
    print_matrix(a);
    prints("=============================================");
    prints("a +. n = ");
    print_matrix(b);
    prints("=============================================");
}
```

Expected output:
test-matrixAdd.out

```
=============================================
n =
[ 1, 2, 4 ]
[ 3, 4, 5 ]

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

=============================================
a +. n =
[ 2, 4, 8 ]
[ 6, 8, 10 ]

=============================================
```

8.4.61. test-matrixIdentity.mx

```
int main(){
    Matrix m;
```

```
    Matrix n;
    Matrix f;
    m = identity(2);
    n = identity(3);
    f = identity(6);
    prints("=============================================");
    prints(" 2x2 Identity Matrix ");
    print_matrix(m);
    prints("=============================================");
    prints(" 3x3 Identity Matrix ");
    print_matrix(n);
    prints("=============================================");
    prints(" 6x6 Identity Matrix ");
    print_matrix(f);
    prints("=============================================");
}
```

Expected output:
test-matrixIdentity.out

```
=============================================
 2x2 Identity Matrix
[ 1, 0 ]
[ 0, 1 ]

=============================================
 3x3 Identity Matrix
[ 1, 0, 0 ]
[ 0, 1, 0 ]
[ 0, 0, 1 ]

=============================================
 6x6 Identity Matrix
[ 1, 0, 0, 0, 0, 0 ]
[ 0, 1, 0, 0, 0, 0 ]
[ 0, 0, 1, 0, 0, 0 ]
[ 0, 0, 0, 1, 0, 0 ]
[ 0, 0, 0, 0, 1, 0 ]
[ 0, 0, 0, 0, 0, 1 ]

=============================================
```

8.4.62. test-matrixMul2.mx

```
int main(){
    Matrix d;
```

```
    Matrix n;
    Matrix a;
    Matrix q;
    Matrix r;
    Matrix s;
    n = [[1,2,3],[4,5,6], [7,8,9]];
    a = [[9,8,7], [6,5,4], [3,2,1]];
    d = a *. n;
    q = [[1,2,3],[4,5,6]];
    r = [[9,8,7], [6,5,4], [3,2,1]];
    s = q *. r;
    prints("============================================");
    prints("Matrix a = ");
    print_matrix(a);
    prints("============================================");
    prints("Matrix n = ");
    print_matrix(n);
    prints("============================================");
    prints("a *. n = ");
    print_matrix(d);
    prints("============================================");
    prints("Matrix q = ");
    print_matrix(q);
    prints("============================================");
    prints("Matrix r = ");
    print_matrix(r);
    prints("============================================");
    prints("q *. r = ");
    print_matrix(s);
    prints("============================================");
}
```

Expected output:
test-matrixMul2.out

```
============================================
Matrix a =
[ 9, 8, 7 ]
[ 6, 5, 4 ]
[ 3, 2, 1 ]

============================================
Matrix n =
[ 1, 2, 3 ]
[ 4, 5, 6 ]
[ 7, 8, 9 ]

============================================
```

```
a *. n =
[ 90, 114, 138 ]
[ 54, 69, 84 ]
[ 18, 24, 30 ]

==========================================
Matrix q =
[ 1, 2, 3 ]
[ 4, 5, 6 ]

==========================================
Matrix r =
[ 9, 8, 7 ]
[ 6, 5, 4 ]
[ 3, 2, 1 ]

==========================================
q *. r =
[ 30, 24, 18 ]
[ 84, 69, 54 ]

==========================================
```

8.4.63. test-matrixMul.mx

```
int main(){
    Matrix d;
    Matrix n;
    Matrix a;
    n = [[1,2],[3,4]];
    a = [[5,6],[7,8]];
    d = a *. n;
    prints("==========================================");
    prints("Matrix a = ");
    print_matrix(a);
    prints("==========================================");
    prints("Matrix n = ");
    print_matrix(n);
    prints("==========================================");
    prints("a *. n = ");
    print_matrix(d);
}
```

Expected output:
test-matrixMul.mx

```
==========================================
Matrix a =
[ 5, 6 ]
[ 7, 8 ]

==========================================
Matrix n =
[ 1, 2 ]
[ 3, 4 ]

==========================================
a *. n =
[ 23, 34 ]
[ 31, 46 ]
```

8.4.64. test-matrix.mx

```
int main(){
    Matrix m;
    Matrix n;
    Matrix f;
    Matrix a;
    Matrix b;
    Matrix c;
    Matrix d;
    Matrix e;
    Matrix g;
    Matrix h;
    Matrix i;
    Matrix j;
    Matrix k;
    Matrix l;
    int colsa;
    int colsn;
    int rows;
    float ab;
    float ac;
    float ad;
    ab = 2.0;
    ac = 3.0;
    ad = ab + ac;
    ad = ab + 5;
    ad = ad + 5;
    m = [[2,3,4],[6,4,5],[7,8,1],[9,7,8]];
    n = [[1,2,4],[3,4,5]];
    a = [[1,2,4],[3,4,5]];
    e = [[2,2],[2,2]];
    f = [[1,2,3,4],[4,5,2,3],[7,5,2,3]];
```

```
    m = m';
    b = a +. n;
    #b = a +. e;
    c = a -. n;
    d = e *. n;
    g = identity(3);
    h = g**.4;
    i = transformation(e, 2);
    colsa = numCols(a);
    colsn = numCols(n);
    rows  = numRows(n);
    print_matrix(m);
    prints("==============================================");
    print_matrix(n);
    prints("==============================================");
    print_matrix(f);
    prints("==============================================");
    #print_matrix(b);
    prints("==============================================");
    print_matrix(c);
    prints("==============================================");
    print_matrix(d);
    prints("==============================================");
    print_matrix(g);
    prints("==============================================");
    print_matrix(h);
    prints("==============================================");
    print_matrix(i);
    prints("==============================================");
    print(colsa);
    prints("==============================================");
    print(colsn);
    prints("==============================================");
    print(rows);
    prints("==============================================");
    rows += 4;
    print(rows);
    prints("==============================================");
    rows -= 2;
    print(rows);
    prints("==============================================");
    rows *= rows;
    print(rows);
    prints("==============================================");
    printf(ad);
    prints("==============================================");
    prints("Done!");
}
```

Expected output:
test-matrix.out

```
[ 2, 6, 7, 9 ]
[ 3, 4, 8, 7 ]
[ 4, 5, 1, 8 ]

================================================
[ 1, 2, 4 ]
[ 3, 4, 5 ]

================================================
[ 1, 2, 3, 4 ]
[ 4, 5, 2, 3 ]
[ 7, 5, 2, 3 ]

================================================
================================================
[ 0, 0, 0 ]
[ 0, 0, 0 ]

================================================
[ 8, 12, 18 ]
[ 8, 12, 18 ]

================================================
[ 1, 0, 0 ]
[ 0, 1, 0 ]
[ 0, 0, 1 ]

================================================
[ 4, 0, 0 ]
[ 0, 4, 0 ]
[ 0, 0, 4 ]

================================================
[ -2, -2 ]
[ -2, -2 ]

================================================
3
================================================
3
================================================
2
================================================
6
================================================
```

```
4
==========================================
16
==========================================
12
==========================================
Done!
```

8.4.65. test-matrixprint.mx

```
int main() {
    Matrix m;

    m = [[4,3], [5,6], [3,2]];
    print_matrix(m);

    m = [[30,90,48], [20,30,18], [39,20,48], [67,38,92]];
    print_matrix(m);
    print_matrix([[0,1,0,3],[0,6,3,2],[4,5,5,3]]);
}
```

Expected output:
test-matrixprint.mx

```
[ 4, 3 ]
[ 5, 6 ]
[ 3, 2 ]

[ 30, 90, 48 ]
[ 20, 30, 18 ]
[ 39, 20, 48 ]
[ 67, 38, 92 ]

[ 0, 1, 0, 3 ]
[ 0, 6, 3, 2 ]
[ 4, 5, 5, 3 ]
```

8.4.66. test-matrixRotation.mx

```
int main(){
    Matrix n;
    n = [[1,2],[3,4]];
    prints("==========================================");
    prints("Matrix n:");
    print_matrix(n);
```

```
    prints("============================================");
    prints("Matrix 90 Degrees Clockwise");
    print_matrix(transformation(n,5));
    prints("============================================");
    prints("Matrix 180 Degrees");
    print_matrix(transformation(n,6));
    prints("============================================");
    prints("Matrix 90 Degrees Anticlockwise:");
    print_matrix(transformation(n,7));
    prints("============================================");
}
```

Expected output:
test-matrixRotation.out

```
============================================
Matrix n:
[ 1, 2 ]
[ 3, 4 ]

============================================
Matrix 90 Degrees Clockwise
[ 3, 4 ]
[ -1, -2 ]

============================================
Matrix 180 Degrees
[ -1, -2 ]
[ -3, -4 ]

============================================
Matrix 90 Degrees Anticlockwise:
[ -3, -4 ]
[ 1, 2 ]

============================================
```

8.4.67. test-matrixRowsColumns.mx

```
int main(){
    Matrix m;
    Matrix n;
    int colsm;
    int rows;
    m = [[2,3,4],[6,4,5],[7,8,1],[9,7,8]];
    n = [[1,2,4],[3,4,5]];
```

```
    colsm = numCols(m);
    rows  = numRows(n);
    prints("=============================================");
    print_matrix(m);
    prints("=============================================");
    prints("Number of columns for m are: ");
    print(colsm);
    prints("=============================================");
    print_matrix(n);
    prints("=============================================");
    prints("Number of rows for n are: ");
    print(rows);
    prints("=============================================");
}
```

Expected output:
test-matrixRowsColumns.out

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

=============================================
Number of columns for m are:
3
=============================================
[ 1, 2, 4 ]
[ 3, 4, 5 ]

=============================================
Number of rows for n are:
2
=============================================
```

8.4.68. test-matrixScalar.mx

```
int main(){
    Matrix d;
    Matrix n;
    Matrix q;
    Matrix s;
    Matrix o;
    Matrix p;
    d = [[1,2,3],[4,5,6], [7,8,9]];
```

```
    n = d**.2;
    q = [[1,2,3],[4,5,6]];
    s = q **.0;
    o = identity(4);
    p = o **.-9;
    prints("=============================================");
    prints("Matrix d = ");
    print_matrix(d);
    prints("=============================================");
    prints("Matrix d**.2 = ");
    print_matrix(n);
    prints("=============================================");
    prints("Matrix q = ");
    print_matrix(q);
    prints("=============================================");
    prints("Matrix q**.0 = ");
    print_matrix(s);
    prints("=============================================");
    prints("Matrix o = ");
    print_matrix(o);
    prints("=============================================");
    prints("Matrix o**.-9 = ");
    print_matrix(p);
    prints("=============================================");
}
```

Expected output:
test-matrixScalar.out

```
=============================================
Matrix d =
[ 1, 2, 3 ]
[ 4, 5, 6 ]
[ 7, 8, 9 ]

=============================================
Matrix d**.2 =
[ 2, 4, 6 ]
[ 8, 10, 12 ]
[ 14, 16, 18 ]

=============================================
Matrix q =
[ 1, 2, 3 ]
[ 4, 5, 6 ]

=============================================
Matrix q**.0 =
```

```
[ 0, 0, 0 ]
[ 0, 0, 0 ]

===========================================
Matrix o =
[ 1, 0, 0, 0 ]
[ 0, 1, 0, 0 ]
[ 0, 0, 1, 0 ]
[ 0, 0, 0, 1 ]

===========================================
Matrix o**.-9 =
[ -9, 0, 0, 0 ]
[ 0, -9, 0, 0 ]
[ 0, 0, -9, 0 ]
[ 0, 0, 0, -9 ]

===========================================
```

8.4.69. test-matrixSub.mx

```
int main(){
    Matrix n;
    Matrix b;
    Matrix a;
    n = [[1,2,4],[3,4,5]];
    a = [[1,2,4],[3,4,5]];
    b = a -. n;
    prints("============================================");
    prints("n = ");
    print_matrix(n);
    prints("============================================");
    prints("a = ");
    print_matrix(a);
    prints("============================================");
    prints("a -. n = ");
    print_matrix(b);
    prints("============================================");
}
```

Expected output:
test-matrixSub.out

```
===========================================
n =
[ 1, 2, 4 ]
```

```
[ 3, 4, 5 ]

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

==========
a -. n =
[ 0, 0, 0 ]
[ 0, 0, 0 ]

==========
```

8.4.70. test-matrixTransformation.mx

```
int main(){
    Matrix n;
    n = [[1,2],[3,4]];
    prints("==========");
    prints("Matrix n:");
    print_matrix(n);
    prints("==========");
    prints("Matrix Transformation About y = x");
    print_matrix(transformation(n,1));
    prints("==========");
    prints("Matrix Transformation About y = -x");
    print_matrix(transformation(n,2));
    prints("==========");
    prints("Matrix Transformation About x-axis");
    print_matrix(transformation(n,3));
    prints("==========");
    prints("Matrix Transformation About y-axis");
    print_matrix(transformation(n,4));
    prints("==========");
}
```

Expected output:
test-matrixTransformation.out

```
==========
Matrix n:
[ 1, 2 ]
[ 3, 4 ]

==========
```

```
Matrix Transformation About y = x
[ 3, 4 ]
[ 1, 2 ]

==========================================
Matrix Transformation About y = -x
[ -3, -4 ]
[ -1, -2 ]

==========================================
Matrix Transformation About x-axis
[ 1, 2 ]
[ -3, -4 ]

==========================================
Matrix Transformation About y-axis
[ -1, -2 ]
[ 3, 4 ]

==========================================
```

8.4.71. test-matrixTranspose.mx

```
int main(){
    Matrix n;
    Matrix a;
    Matrix q;
    Matrix r;
    n = [[1,2],[4,5]];
    a = [[9,8,7], [6,5,4]];
    q = n';
    r = a';
    prints("==========================================");
    prints("Matrix a = ");
    print_matrix(a);
    prints("==========================================");
    prints("Matrix a' = ");
    print_matrix(r);
    prints("==========================================");
    prints("Matrix n = ");
    print_matrix(n);
    prints("==========================================");
    prints("Matrix n' = ");
    print_matrix(q);
    prints("==========================================");
}
```

Expected output:
test-matrixTranspose.out

```
================================================
Matrix a =
[ 9, 8, 7 ]
[ 6, 5, 4 ]

================================================
Matrix a' =
[ 9, 6 ]
[ 8, 5 ]
[ 7, 4 ]

================================================
Matrix n =
[ 1, 2 ]
[ 4, 5 ]

================================================
Matrix n' =
[ 1, 4 ]
[ 2, 5 ]

================================================
```

8.4.72. test-relationalops.mx

```
int main()
{
    printb(10 > 0);
    printb(10 > -10);
    printb(10 > (3 + -4));
    printb(-10 > -20);
    printb(-10 > -10);

    prints(">>>>>>");

    printb(0 < 10);
    printb(-10 < 10);
    printb((-3 + 4) < 10);
    printb(-20 < -10);
    printb(-10 < -10);

    prints(">>>>>>");

    printb(5 <= 10);
```

```
    printb(5 <= 5);
    printb(-1 >= -1);
    printb(0 >= 0);
    printb(12 >= -3);

    prints(">>>>>>");

    printb("h" == "h");
    printb(23 == 24);
    printb(true == false);
    printb(false != false);
    printb("he" != "He");
}
```

Expected output:
test-relationalops.out

```
1
1
1
1
0
>>>>>>
1
1
1
1
0
>>>>>>
1
1
1
1
1
>>>>>>
1
0
0
0
1
```

8.4.73. test-sampleprogram1.mx

```
int main() {
    Matrix m;
    Matrix n;
```

```
    m = [[2,4],[6,8],[10,12]];
    n = m';
    print_matrix(m);
    print_matrix(n);
    return 0;
}
```

Expected output:
test-sampleprogram1.out

```
[ 2, 4 ]
[ 6, 8 ]
[ 10, 12 ]

[ 2, 6, 10 ]
[ 4, 8, 12 ]
```

8.4.74. test-sampleprogram2.mx

```
int main() {
    Matrix m1;
    Matrix m2;

    m1 = [[1,2],[3,4]];
    m2 = [[2,4],[6,8]];

    print_matrix(m1 +. m2);
    print_matrix(m1 -. m2);
    print_matrix(m1 *. m2);
    print_matrix(m2 **. 3);
}
```

Expected output:
test-sampleprogram2.out

```
[ 3, 6 ]
[ 9, 12 ]

[ -1, -2 ]
[ -3, -4 ]

[ 14, 20 ]
[ 30, 44 ]
```

```
[ 6, 12 ]
[ 18, 24 ]
```

8.4.75. test-transpose1.mx

```
int main() {
    Matrix n;

    n = [[1,2,3],[4,5,6]]';

    print_matrix(n);
}
```

Expected output:
test-transpose1.out

```
[ 1, 4 ]
[ 2, 5 ]
[ 3, 6 ]
```

8.4.76. test-while.mx

```
int main() {
    int a;
    int b;
    int c;
    a = 0;
    b = 3;
    c = 5;

    prints("============================================");
    prints("while( a < 10)");
    while(a < 10) {
        if( a == 5) {
            prints("a is equal to 5");
        } else {
            print(a);
        }
        a = a + 1;
    }

    prints("============================================");
    prints("c = 5 and b = 3");
    prints("while( c > b)");
    while( c > b) {
```

```
        print(c);
        c = c - 1;
    }

}
```

Expected output:
test-while.out

```
================================================
while( a < 10)
0
1
2
3
4
a is equal to 5
6
7
8
9
================================================
c = 5 and b = 3
while( c > b)
5
4
```