Matcat

A High-Level Programming Language for computing Linear Algebra

Mariam Khmaladze Davit Barblishvili James Ryan Andreas Cheng

System Architect Language Guru Tester Manager

mk4069@barnard.edu, {db3230, james.ryan, hc3142}@columbia.edu

April 26, 2021

Contents

1 Introduction

1.1 Background

Matcat is an imperative programming language designed for the purpose of writing and computing linear algebra operations. This language will ship with a standard library of common matrix operations (see 7.2) in order to provide a core set of tools around which a Matcat programmer can operate. These matrix operations were largely influenced by the Wolfram Language (see Wolfram: Matrices and Linear Algebra) and thus aims to offer similar value.

1.2 Goals

- ➢ Simplify matrix operations. With a matrix type and a standard library with which to manipulate it, we aim to save programmers time and provide a solid foundation for more advanced problem solving.
- ➢ To provide Java or C-like syntax. By working with these norms of popular imperative languages, we hope to shorten our language's learning curve.
- ➢ To provide consistent and intuitive operations across different types. Since matrices share many math operations with int and double types, we prioritized making sure the code snippet

```
a+b
```

  works as expected whether a and b are of type matrix, int or double.

- ➢ Optimize compile time with static typing.

1.3 Type System Overview

Matcat is strongly typed and provides a static type scope. A variable must both be declared before it is used and contain only values of the type with which it is declared. The textual structure of a program - its nest of encapsulating braces - determines the scope of a variable.

2 Language Tutorial

2.1 Setting up the Environment

Matcat comes with a Dockerfile in order to attain all language dependencies. While creating a local image is by far the quickest way to get started:

```
docker build . -t somename
docker run --rm -it -v {`pwd`}:/home/matcat -w=/home/matcat somename
```

Or, one can use the Dockerfile commands as shown in 10.11 Dockerfile as a reference list of package dependencies and their versions.

2.2 Downloading and Building Matcat

The following terminal commands:

```
git clone https://github.com/davitbarblishvili/Matcat
cd Matcat
make
```

will download and compile all necessary features for immediate use.

The command:

```
./testall.sh
```

will run our full array of testing, letting one know if all dependencies have been acquired and packages have compiled properly.

2.3 Writing and Compiling a Simple Matcat Program

The following code implements and uses the GCD algorithm.

```
func gcd(int a, int b) int {
      while(a != b) {
            if (a < b) {
                  b = b - a;
      }
             else {
                  a = a - b;
             }
      }
      return a;
}
func main() int {
      int x = 10;
      int y = 5;
      int g = gcd(x, y);
      print(g);
      return 0;
}
```

Store this code in a file named gcd.mc and run the command

```
./run.sh gcd.mc
...
5
```

to run all compilation steps and the resulting executable.

3 Language Reference Manual

3.1 Lexical Conventions

3.1.1 Comments

Matcat has single-line and multi-line comments. Both types of comments do not nest.

All tokens on the line that starts with `//` are part of a single-line comment which are not parsed.

```
// TODO: find column space of m
m = [[1,2,3],[4,5,6],[7,8,9]]
```

Multi-line comments begin with `/*` and end with `*/`.

```
/* Eigenvalues are a set of scalars.
 * We can transform them as eigenvectors.
 *
 *    <- Asterisks between /* and */ are optional.
 */
```

3.1.2 Identifiers

There are certain rules for defining a valid Matcat identifier. These rules must be followed. Otherwise, a compile-time error arises:

- The only allowed characters for identifiers are all alphanumeric characters([A-Z],[a-z],[0-9]), and '_' (underscore).
- Identifiers should not start with digits([0-9]). For example, "123cat" is not a valid Matcat identifier.
- Matcat identifiers are case sensitive.
- There is no limit on the length of the identifier, but it is advisable to use an optimum length of 3–15 characters only.
- Reserved words can't be used as an identifier. For example, "int matrix = 20;" is an invalid statement as 'matrix' is a reserved word (see 3.4).

```
// Valid Identifiers
matCat;
thisIsValid;
this_is_valid_too;

//Invalid Identifiers
Mat Cat; //contains space
123cat;  //starts with a digit
2;       // includes only a digit
inverse; //reserved keyword in Matcat
```

3.1.3 Operators

Matcat utilizes following reserved operators:

```
- +  x * / ** <  >  <=  >=  !=  == || && cr dot
```

3.1.4 Keywords

Matcat has a set of keywords that are reserved and cannot be used as variables, methods, classes, or any other identifiers:

```
while for if else return true false int double bool matrix print
printd printStr printm transpose inverse dim det rref cr dot
```

3.1.5 Grouping and Code Blocks

An expression such as x = 0 or x = x + 1 or print(...) becomes a statement when it is followed by a semicolon, as in:

```
 x = 0;
 x = x + 1;
 print(x);
```

In Matcat, the semicolon is a statement terminator rather than a separator as it is in languages like Ocaml. Braces { and } are used to group declarations and statements together into a compound statement or block so that they are syntactically equivalent to a single statement. The braces that surround the statements of a function are one obvious example. Braces around multiple statements after an if, else, while, or for are another. There is no semicolon after the right brace that ends a block.

3.1.6 Separators

Matcat uses parentheses to override the default precedence of expression evaluation. To separate expressions and denote control flow Matcat is utilizing ';', {} respectively. It is not possible in Matcat to separate two consecutive expressions on the same line. However, it is possible to both declare multiple variables in one line separated by commas, and to assign one value to multiple variables:

```
int myInt = (2+3)*5; // overrides default operator precedence
int a, b, c;         // declaring multiple variables
a = b = c = 5;       // assigning 5 to a, b, and c
```

3.1.7 Literals

Literals represent strings or one of Matcat's built-in data types: double, string, int, boolean, matrix. Thus all of the following productions are valid expressions:

expr → IntLit

→ DoubleLit

→ BoolLit

→ StringLit

→ MatrixLit

3.1.7.1 Double Literals

A float literal is a number with a whole number, an optional decimal point, a fraction, and an exponent. The regex is shown below:

```
digits '.'  digit* ( ['e' 'E'] ['+' '-']? digits )?
```

Examples of double literals:

```
10
1.2
14.
0.00005
1e+2
2.3E-3
```

3.1.7.2 String Literals

A string literal is a sequence of characters enclosed in single or double quotation marks, i.e. abcdefghijklmnopqrstuvwxyz. The matching regex is

`("[^"'\\]*(\\.[^"\\]*)*")|('[^"'\\]*(\\.[^'\\]*)*')`

Example of string literals:

```
"This language is written in Ocaml"
"A string literal"
```

3.1.7.3 Int Literals

An integer literal is any sequence of integers between 0 and 9. The matching regex is `[0-9]+`.

3.1.7.4 Boolean Literals

Boolean types represent true and false. They are represented in Matcat by the **`true`** and **`false`**

keywords.

3.1.7.5 Matrix Literal

Matrix literal is a multi-dimensional array i.e, one array consisting of multiple subarrays - all the arrays within a matrix are treated as matrix literals themselves. All of the elements within a matrix are stored as float types in order to be most flexible and explicit at compile time. For a comprehensive set of detailed matrix behavior, see 7.3 Test Cases.

```
matrix mt = [[2,3,4], [5,6,7]]; // matrix initialization
```

3.2 Data Types

Matcat represents all pieces of data as either an object or a primitive. There are only a few basic data types in the language.

3.2.1 Primitives

Primitives are series of bytes of some fixed length, and there are four primitive types in Matcat:

- int (4 bytes, 2's complement)
- double (8 bytes, IEEE standard double)
- char (1 byte, ASCII; a single byte, capable of holding one character in the local character set)
- bool (1 byte; 00000001 for true, 00000000 for false)

  * Note that matrix is not a primitive data type.

3.2.2 Objects

Matrix is the single object type in Matcat, as custom user defined types are not supported. References are completely under the hood and not user-accessible as pointers.

3.2.3 Mutability

The primitive types in Matcat are immutable, including int, double, and boolean. Strings are not primitives, but are also immutable. All operations which modify these types actually return new objects. All types defined in the standard library - namely matrix - are mutable in the sense that modifying them does not overwrite the underlying object. Assigning to any variable will still overwrite its underlying object, i.e. x = 3; x = 4 assigns an integer literal with value 3 to the variable x, and then assigns a different integer literal with the value 4 to the same variable. Likewise, matrix x = [[1, 2, 3]], x = [[4, 5, 6]] will assign one object of type list to x, and then assign a different object to it. On the other hand, x = [[1, 2, 3]], x[0][1] = 4 will modify the underlying data associated with the variable x, returning x = [[1, 4, 3]].

3.2.4 Memory Model

The Matcat language will be "pass by reference," as in Java. We are going to implement a memory management system that will be handled internally by a simple garbage collector - objects that no longer have references attached to them deallocated. No explicit memory management is required.

3.3 The Matcat Type System

3.3.1 Overview

The Matcat language uses static typing, meaning all data types need to be set properly at compilation. This guarantees that all the errors will be caught during the compilation stage, and that runtime will be error-free.

3.3.2 Explicit Typing

Explicitly typed variables are denoted by a type like an int, double, boolean, matrix, followed by a variable name and semicolon.

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

Once an identifier has been associated with a class, it cannot be assigned to an object of a different class.

```
int x = 3;
x='s';          //this is not allowed
x=4;           //this is allowed
double y=3.0;
```

3.3.3 Optimization

The compiler infers that a piece of data will consistently be of a certain primitive type, unboxes the data, and converts it to a primitive type. Otherwise, it fails at compile-time. This means that runtime performance will be as optimized as in C and Java.

3.3.4 Function Typing

Functions in Matcat are declared explicitly. The types of function calls will be checked at a compile-time and will be compiled to efficient machine code. A function is denoted by keyword func, followed by the name of the function, list of arguments, return type, and body.

```
func myFunction(dataType x, dataType y) returnType {}
```

Arguments and return types of the function must be explicitly typed upon declaration. Matcat allows the return of multiple variables of different data types. Examples will be shown below.

3.4 Statements and Expressions

A Matcat program consists of a series of statements, such as the following:

- Expression
- Class Declaration
- Function Definition
- Return Statement
- If-else statement
- For loop
- While loop

Statements are encapsulated together to form blocks by existing within the same {}.

3.4.1 If-else statements

An if statement consists of a condition - an expression which evaluates to a Boolean type - and a block of statements which execute if this condition evaluates to true. An else statement optionally follows an if statement, and its block of statements execute when it is paired if the statement's condition evaluates to false.

```
if(condition) {
   // executes when the condition is true
} else {
   // executes when the condition is false
}
```

3.4.2 While Loops

A While loop consists of a condition and a series of statements. These statements execute until the condition evaluates to false.

```
while(condition) {
   // executes until condition is false
}
```

3.4.3 For Loops

A For statement consists of three parts:

1. An initial expression in which variable assignment usually takes place. The type of this variable must be explicitly declared prior to the loop.

2. A condition that must evaluate to true in order to execute the corresponding block of statements.
3. An expression which executes after each loop takes place; typically an increment.

```
int i;
for(i = 0; i < N; i++) {
   // series of statements
}
```

3.4.4 Expressions and Operators

Expressions consist of statements that are chained together using operators. All expressions, even assignments, evaluate to a built-in or library defined data type.

3.4.4.1 Unary Operators

Unary operators act on a single variable or expression. These include:

1. Negation

```
int x = -2;
```

2. Boolean Flipping

```
bool a = false;
bool b = !a; // b is true
```

3.4.4.2 Binary Operators

Binary operators take the value of the expression immediately to its right and left and reconcile them to produce the value of the overall expression. Both of these expressions must evaluate to legal types for the given operator, detailed below. These follow the form

| Expression * Operator * Expression

where * can represent any of the following operators:

1. Assignment Operator

The assignment operator stores values in variables, with the value on the right stored in the variable on the left. This can be applied to any datatype.

```
double s = 7.2;
```

2. Arithmetic Operator
    a. Addition

```
2 + 8 // evaluates to 10
[[1, 0]] + [[0, 1]] // evaluates to [[1, 1]]
```

    b. Subtraction

```
2 - 8 // evaluates to -6
[[5, 5]] - [[4, 2]] // evaluates to [[1, 3]]
```

    c. Multiplication

```
3 * 4 // evaluates to 12
[[1, 2], [4, 5]] * [[1, 2], [4, 5]] //evaluates to [[9, 12],[22, 33]]
```

    d. Division

```
9 / 2 // evaluates to double type 4.5
```

    e. Exponentiation

```
3 ** 2 // evaluates to 9
```

3. Matrix Specific Operators
    a. Dot Product

```
[[2, 3]] dot [[3, 4]] // evaluates to 18
[[1,2],[3,2]] dot [[2,3],[2,1]] // evaluates to 16
```

4. Relational Operators

```
int x = 1;
int y = 2;
bool b = x < y; // b holds true
b = x == y; // b holds false
b = x != y; // b holds true
```

3.4.5 Operator Precedence

The table below shows operator precedence, starting from the lowest precedence and increasing down the table.

Operator	Meaning	Associativity
;	Sequencing	Left
=	Assignment	Right
.	Access	Left
||	Logical Or	Left
&&	Logical And	Left
==, !=	Equality, Inequality	Left
<, >, <=, >=	Comparisons	Left
+, -	Addition, Subtraction	Left
*, /	Multiplication, Division	Left
dot	Dot Product	Left
**	Exponentiation	Right
!	Logical Not	Right

3.4.6 Functions

A function is a type of statement. It takes a list of arguments and returns a list of values. Both arguments and return values must have explicit types and must be passed and caught in the order given in the function definition. The syntax is shown below.

```
func getSum(int one, int two) int {
   return one + two;
}
```

Function getSum expects two integer arguments, and returns one integer argument as stated by the datatype after the ().

3.4.7 Function Calls

Functions are called using their identifier and passing legal arguments within the (). Return values can be assigned to newly declared or previously declared variables.

```
int n;
n = getSum(5, 4); // n holds 9
```

3.5 Standard Library

3.5.1 Print Functions

Matcat comes with a variety of print functions, each tailored to a specific data type. Whitespaces in strings are not ignored.

- print: takes an integer argument and prints this to stdout.
- printb: takes a boolean argument and prints this to stdout.
- printd: takes a double argument and prints this to stdout.
- printStr: takes a string literal argument and prints this to stdout.
- printm: takes a matrix argument and pretty prints this to stdout.

3.5.2 Matrix Library Functions

The Matcat language will use some of the library functions that will be accessible to the user for matrix manipulation. The following is an exhaustive list of such functions, beginning with a notation example:

- *funcIdentifier(argumentTypes): returnType ; description of functionality*
- transpose(matrix): matrix ; find transpose of matrix
- inv(matrix): matrix ; find inverse of matrix
- isInv(matrix): bool ; return whether a matrix is invertible
- det(matrix): int ; find determinant of matrix
- rref(matrix): find matrix's reduced row echelon form
- rotate90(matrix): matrix ; rotate all matrix values 90 degrees
- check_symmetry(matrix): bool ; return whether a matrix is symmetrical
- dim(): takes a matrix as an argument and returns two integers describing the dimension of the matrix

3.5.3 Matrix

Matcat provides an out-of-the-box matrix data type. The elements of a matrix are themselves matrices, i.e. m[i] evaluates to matrix type if this index is inbound. The elements of a matrix are stored internally as double types, but have very specific rules for initialization: all values passed into a matrix at initialization must be of the same type. These key attributes of matrix are outlined in code below.

```
matrix m = [[1, 2, 3], [4, 5, 6 ]];
m[0] // evaluates to [[1, 2, 3]] of type matrix
m[0][0] // evaluates to 1

matrix ok = [[1.3, 5.5, 9.0]]; // legal, all doubles
matrix bad = [[1, 2, 3.5]]; // not legal due to more than one type
(int and double)
```

The following operations are supported:

Method/operator	Behavior
m[i][j]	Returns the j-th element in the i-th row

m[i,:]	Returns i'th row as matrix
m[:i,]	Returns i'th column as matrix
m[:,:]	Returns the main diagonal of the squared matrix
m1 + m2	Returns the matrix sum of m1 and m2
m1 - m2	Returns the matrix difference of m1 and m2
m1 * m2	Returns the matrix with matrix multiplication performed
m1 / m2	Returns matrix after division
m1 == m2	Returns true if m1 is identical to m2. Otherwise, return false.
m^n	Returns matrix with each element raised to the n'th power
m dot r	Returns dot product of two matrices; performance is not defined in the case of matrix with dimensionality > 1

3.6 Sample Code

```
func main() int{
    matrix m;
    matrix c;
    matrix t;
    double d;
    matrix rhs;
    matrix lhs;
    double dp;
    matrix p;

    m = [[1,2,-1],[-2,0,1],[1,-1,0]];
    rhs = [[1,2,3,4]];
    lhs = [[2,5,7,5]];
    printStr("matrix m:");
    printm(m);
```

```
    //invertible
    isInv(m);

    c=inv(m);
    printStr("inverse of m is c:");
    printm(c);

    t=transpose(m);
    printStr("transpose of m is t:");
    printm(t);

    d=det(m);
    printStr("determinant of m is");
    printd(d);


    dp=rhs dot lhs;
    printStr("dot product of");
    printm(rhs);
    printStr("and");
    printm(lhs);
    printd(dp);


    p=m^5;
    printStr("m in power 5 is:");
    printm(p);


    rotate90(m);
    printStr("m rotated by 90 is:");
    printm(m);

    return 0;
}
```

4 Project Plan

4.1 Project Processes

4.1.1 Planning

We had team meetings usually twice a week. One was on Wednesday and the other one was on Saturday/Sunday, which is also the Office Hour of our assigned TA. These meetings were an opportunity to ask questions and receive feedback.

Our communication happened mostly over Slack. This allowed us to work efficiently. We attempted to categorize different conversations in different slack channels, such as #coding, #general, #testing, and #useful-materials to help tracking different conversations and share our knowledge.

Other than the Slack workspace itself, we also installed Slack Apps like Workstreams that work as a kanban board to visualize the work of each group member.

4.1.2 Specification

The main specification is based on the LRM. It is sometimes modified because of time or technical constraints as we know more about building a compiler from the lectures and in the Development and Testing phase.

4.1.3 Development

We mostly use Visual Studio Code for development. We make contributions to a GitHub repo to collaboratively code Matcat. We create new branches for features to implement different features in parallel. We merge the changes to the main branch and run the integration test afterward to make sure the main branch always functions as we expected.

4.1.4 Testing

Integration test is adopted to debug and validate different git merges. At the beginning of the project, we mostly rely on the testall.sh provided in the MicroC directory. As time went by, we realized that we better have a script that is easier to maintain and can print more pretty output to work more efficiently. So we write another one in Python.

4.2 Programming Style Guide

1. For Ocaml, indent with spaces.
2. For Ocaml, indentation size: 2 spaces.
3. For C, try to follow the Linux kernel coding style if possible.
4. Keep lines under 120 characters with rare exceptions
5. Write comment to explain complicated code

6. Make sure the code is compilable before commiting/merging to the main branch
7. If changes made are across different files and they are related, try commit all of these files in one commit for easier code reviews/debuggings
8. Commit frequently

4.3 Software Development Environment and Tools

Languages: OCaml, Shell script, and Python

IDE: Visual Studio Code

Text editors: vim, Sublime

Environment: OSX 10.15 and Ubuntu 18.14.X.

Version Control System: Github-hosted Git repository

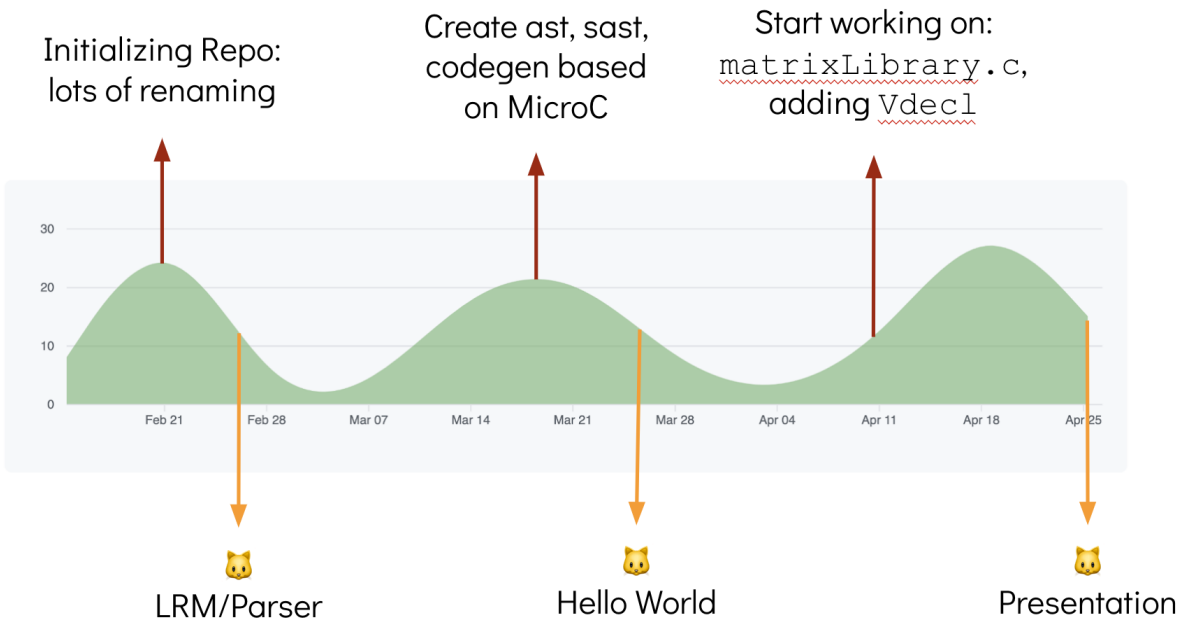
4.4 Roles and Responsibilities

Team Member	**Main Responsibility**
Davit	● Implemented polymorphism for all the operators that accepts Matrix in addition to non-primitive data types; ● Semantically checked data types for built-in functions and operands for the operator; ● Implemented declarations for built-in functions in Codegen. ● Created a data type Matrix that is implemented as 2D array in C ● Successfully attached data types returned from C library to Matcat data types; ● Created test cases to check built-in functions; ● Contributed to Slides
Mariam	Matrix, matrix operations, matrixLibrary
James	Generating test cases, Final Report, writing and verifying the LRM
Andreas	Making variable declaration as a statement available; Interpreter-like feature (to get rid of main, but we failed); Python automatic test suite; Final Report, Slides Design; Auto signing script

4.5 Project Timeline

Date	**Milestone/Works**
Jan 18	Initial Discussion to decide language

Feb 3	Language Proposal
Feb 24	Initial Parser
Feb 24	Language Reference Manual
Mar 24	First Program (Hello World) Demo
Time between Mar 24 - Apr 26	Most developments happen in this time range
Apr 26	Final Presentation and Report



Detail project git log can be found in [10.1 Project Log](#).

5 Language Evolution

5.1 Original Thoughts

In the proposal stage, we decided to design a C-like language that makes Linear Algebra easier through supporting matrix operations such as dot product and cross product natively. In the sense that we are implementing a matrix, we felt like it is better to implement vectors as well.

5.2 Spicing up the Language

We were told that our language seems too easy. So we designed more features in Matcat. The features are listed as follow:

1. A function that returns multiple values
2. Support for user defined types as Structs
3. Support computing eigenvalues and eigenvectors in the standard library
4. Apply row reduction on matrix
5. Make running code seems like an interpreter (The main function is not required)

5.3 Narrowing Down the Scope

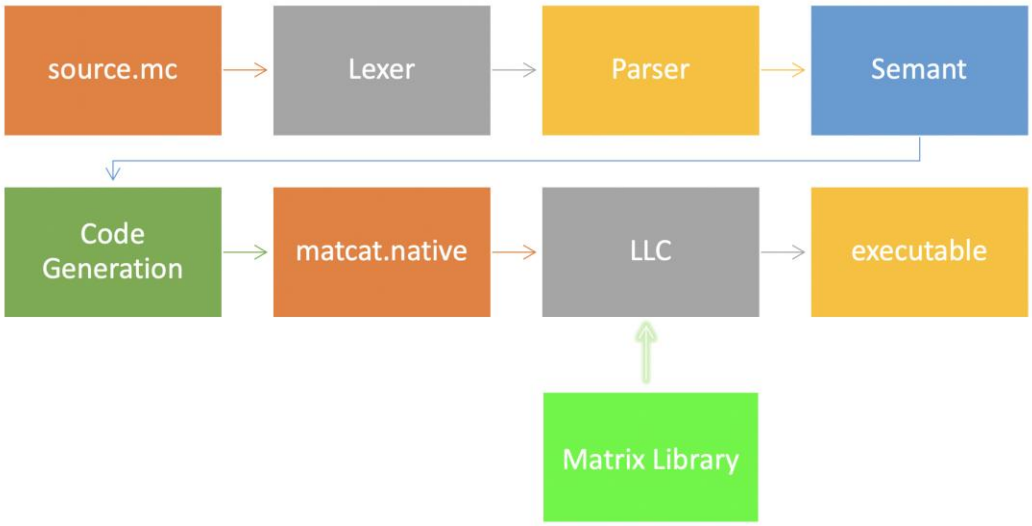
It turns out that many of the features above are not easy to implement at all. So we decided to narrow down the scope because of the time constraint. We also realized that some features are not that common in other languages and might perhaps not have any particular practical use cases, so we decided to get rid of them. We discovered that although llvm do support returning multiple values from a function, the implementation of it is quite challenging and the potential use cases were not valuable enough to justify effort towards this initial goal. Although running Matcat like an interpreter is an appealing idea and makes it resemble Matlab, a similar language, we opted to keep C-like syntax as these conventions should be comfortable for most programmers. Finally, structs were deemed not applicable enough for this kind of math language; perhaps a graph language would want vectors and edges, or data structures trees nodes, but linear algebra is fairly encapsulated by the matrix type

5.4 Design Summary

We overall aimed to provide only the features which a user of a linear algebra type language would need regularly. While returning multiple values from a function is interesting syntactically, both at function declaration and when calling the function, this feature is unnecessary given our language's use case. Thus, with many C syntactic conventions, a programmer should be able to think in typical imperative style.

6 Compiler Architecture

6.1 Block Diagrams of Major Components



6.2 Interfaces Between the Components

The Lexer takes in a .mc source file and tokenizes the input. Errors are caught at this level if illegal symbols are provided (see 3.1.2 for identifier naming conventions). The Parser enforces unambiguous grammar on these tokens, producing an AST if the tokens are sequenced in a valid way. Errors are caught here when major syntactic conventions are neglected - else with no if, etc. Semant recursively traverses the AST, resulting in a semantically checked AST. This compilation step catches errors such as variables being accessed prior to initialization, and other issues of typing and scoping. Codegen uses the SAST as a valid program for which to produce LLVM code; it does this by a traversal of the SAST, linking each part to an LLVM module. *Matrix Library* is C-based code composed of built-in functions for Matcat. Additionally, a Matrix - a special type in Matcat - is assembled in the Matrix Library as well. To implement some functions for matrix manipulations, we used a math library (-lm). To successfully compile Matrix Library, Clang compiler has been used. Here, LLVM and Matrix Library are reconciled and used to run the program.

6.3 Division of Labour

The division of labour can be found in 4.4 Roles and Responsibilities.

7 Test Plan and Scripts

7.1 Representative Source Language Programs

7.1.1 example-1.mc: Returning Matrix from user-defined function in a infinity loop

We chose this because this example can show most of our features within a short scope. You can see that variable declaration is allowed anywhere. Users can define a function that returns a matrix and put the integer formal argument into the matrix element. Other than that, this code also shows our operator polymorphism feature in integer addition and matrix addition using the same plus symbol.

```
func m(int ans) matrix {
        return [[ans, 0],[0, ans]];
}

func main() int {
        printStr("I can only show you the door.");
        int a = 657 + 64;
        while (true) {
                print(a);
                printm(inv( m(42) + m(42) ));
        }
}
```

7.1.2 example-1.ll

```
; ModuleID = 'MatCat'
source_filename = "MatCat"

%struct.matrix = type { i32, i32, float**, 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"%g\0A\00", align 1
@fmt.2 = private unnamed_addr constant [4 x i8] c"%s\0A\00", align 1
@tmp = private unnamed_addr constant [30 x i8] c"I can only show you the door.\00", align 1
@fmt.3 = private unnamed_addr constant [4 x i8] c"%d\0A\00", align 1
@fmt.4 = private unnamed_addr constant [4 x i8] c"%g\0A\00", align 1
@fmt.5 = private unnamed_addr constant [4 x i8] c"%s\0A\00", align 1

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

declare i32 @printMatrix(%struct.matrix*)

declare %struct.matrix* @initMatrix_helper(i32, i32)

declare %struct.matrix* @storeEntries(%struct.matrix*, i32)

declare %struct.matrix* @matrxAdd(%struct.matrix*, %struct.matrix*)

declare %struct.matrix* @matrxSub(%struct.matrix*, %struct.matrix*)
```

```
declare %struct.matrix* @transpose(%struct.matrix*)

declare %struct.matrix* @matrxMult(%struct.matrix*, %struct.matrix*)

declare double @det(%struct.matrix*)

declare %struct.matrix* @inv(%struct.matrix*)

declare double @dot(%struct.matrix*, %struct.matrix*)

declare %struct.matrix* @scaleMatrix(%struct.matrix*, i32)

declare %struct.matrix* @scaleMatrixDouble(%struct.matrix*, double)

declare %struct.matrix* @scalarDivMatrix(%struct.matrix*, i32)

declare %struct.matrix* @scalarDivDoubleMatrix(%struct.matrix*, double)

declare %struct.matrix* @rref(%struct.matrix*)

declare i1 @isInv(%struct.matrix*)

declare %struct.matrix* @accessMatrix(%struct.matrix*, i32, i32)

declare %struct.matrix* @accessMatrix1D(%struct.matrix*, i32)

declare %struct.matrix* @accessMatrixCol(%struct.matrix*, i32)

declare %struct.matrix* @get_diagonal(%struct.matrix*)

declare %struct.matrix* @power_matrix(%struct.matrix*, i32)

declare %struct.matrix* @rotate90(%struct.matrix*)

declare i32 @check_symmetry(%struct.matrix*)

define i32 @main() {
entry:
  %a = alloca i32, align 4
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]*
@fmt.2, i32 0, i32 0), i8* getelementptr inbounds ([30 x i8], [30 x i8]* @tmp, i32 0, i32
0))
  %a1 = alloca i32, align 4
  store i32 721, i32* %a1, align 4
  br label %while

while:                                            ; preds = %while_body, %entry
  br i1 true, label %while_body, label %merge

while_body:                                       ; preds = %while
  %a2 = load i32, i32* %a1, align 4
  %printf3 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]*
```

```
@fmt, i32 0, i32 0), i32 %a2)
  %m_result = call %struct.matrix* @m(i32 42)
  %m_result4 = call %struct.matrix* @m(i32 42)
  %matrxAdd = call %struct.matrix* @matrxAdd(%struct.matrix* %m_result, %struct.matrix*
%m_result4)
  %inv = call %struct.matrix* @inv(%struct.matrix* %matrxAdd)
  %printm = call i32 @printMatrix(%struct.matrix* %inv)
  br label %while

merge:                                            ; preds = %while
  ret i32 0
}

define %struct.matrix* @m(i32 %ans) {
entry:
  %ans1 = alloca i32, align 4
  store i32 %ans, i32* %ans1, align 4
  %ans2 = load i32, i32* %ans1, align 4
  %ans3 = load i32, i32* %ans1, align 4
  %matrix_init = call %struct.matrix* @initMatrix_helper(i32 2, i32 2)
  %storeEntries = call %struct.matrix* @storeEntries(%struct.matrix* %matrix_init, i32
%ans3)
  %storeEntries4 = call %struct.matrix* @storeEntries(%struct.matrix* %matrix_init, i32 0)
  %storeEntries5 = call %struct.matrix* @storeEntries(%struct.matrix* %matrix_init, i32 0)
  %storeEntries6 = call %struct.matrix* @storeEntries(%struct.matrix* %matrix_init, i32
%ans2)
  ret %struct.matrix* %matrix_init
}
```

7.1.3 example-2.mc: Taking Matrix as formal argument and Return it back to main

We chose this to show that we can also take matrix as a formal argument. In this sense, the users can actually create their own matrix library in Matcat, but of course, having a built-in matrix library out of the box is much more convenient for users to manipulate matrices. Users can then focus on coding abstracted matrix programs instead of worrying about those fundamental operations.

```
func add_2_self(matrix abc) matrix {
        return abc+abc;
}
func main() int {
        matrix a = [[1,0],[0,1]];
        printm(add_2_self(a));
}
```

7.1.4 example-2.ll

```
; ModuleID = 'MatCat'
source_filename = "MatCat"

%struct.matrix = type { i32, i32, float**, 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"%g\0A\00", align 1
@fmt.2 = private unnamed_addr constant [4 x i8] c"%s\0A\00", align 1
@fmt.3 = private unnamed_addr constant [4 x i8] c"%d\0A\00", align 1
@fmt.4 = private unnamed_addr constant [4 x i8] c"%g\0A\00", align 1
@fmt.5 = private unnamed_addr constant [4 x i8] c"%s\0A\00", align 1

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

declare i32 @printMatrix(%struct.matrix*)

declare %struct.matrix* @initMatrix_helper(i32, i32)

declare %struct.matrix* @storeEntries(%struct.matrix*, i32)

declare %struct.matrix* @matrxAdd(%struct.matrix*, %struct.matrix*)

declare %struct.matrix* @matrxSub(%struct.matrix*, %struct.matrix*)

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

declare %struct.matrix* @matrxMult(%struct.matrix*, %struct.matrix*)

declare double @det(%struct.matrix*)

declare %struct.matrix* @inv(%struct.matrix*)

declare double @dot(%struct.matrix*, %struct.matrix*)
```

```
declare %struct.matrix* @scaleMatrix(%struct.matrix*, i32)

declare %struct.matrix* @scaleMatrixDouble(%struct.matrix*, double)

declare %struct.matrix* @scalarDivMatrix(%struct.matrix*, i32)

declare %struct.matrix* @scalarDivDoubleMatrix(%struct.matrix*, double)

declare %struct.matrix* @rref(%struct.matrix*)

declare i1 @isInv(%struct.matrix*)

declare %struct.matrix* @accessMatrix(%struct.matrix*, i32, i32)

declare %struct.matrix* @accessMatrix1D(%struct.matrix*, i32)

declare %struct.matrix* @accessMatrixCol(%struct.matrix*, i32)

declare %struct.matrix* @get_diagonal(%struct.matrix*)

declare %struct.matrix* @power_matrix(%struct.matrix*, i32)

declare %struct.matrix* @rotate90(%struct.matrix*)

declare i32 @check_symmetry(%struct.matrix*)

define i32 @main() {
entry:
  %a = alloca %struct.matrix*, align 8
  %a1 = alloca %struct.matrix*, align 8
  %matrix_init = call %struct.matrix* @initMatrix_helper(i32 2, i32 2)
  %storeEntries = call %struct.matrix* @storeEntries(%struct.matrix* %matrix_init, i32 1)
  %storeEntries2 = call %struct.matrix* @storeEntries(%struct.matrix* %matrix_init, i32 0)
  %storeEntries3 = call %struct.matrix* @storeEntries(%struct.matrix* %matrix_init, i32 0)
  %storeEntries4 = call %struct.matrix* @storeEntries(%struct.matrix* %matrix_init, i32 1)
  store %struct.matrix* %matrix_init, %struct.matrix** %a1, align 8
  %a5 = load %struct.matrix*, %struct.matrix** %a1, align 8
  %add_2_self_result = call %struct.matrix* @add_2_self(%struct.matrix* %a5)
  %printm = call i32 @printMatrix(%struct.matrix* %add_2_self_result)
  ret i32 0
}

define %struct.matrix* @add_2_self(%struct.matrix* %abc) {
entry:
  %abc1 = alloca %struct.matrix*, align 8
  store %struct.matrix* %abc, %struct.matrix** %abc1, align 8
  %abc2 = load %struct.matrix*, %struct.matrix** %abc1, align 8
  %abc3 = load %struct.matrix*, %struct.matrix** %abc1, align 8
  %matrxAdd = call %struct.matrix* @matrxAdd(%struct.matrix* %abc2, %struct.matrix* %abc3)
  ret %struct.matrix* %matrxAdd
}
```

More test cases can be found in 10.1 Test Cases with Color, 10.19 Integration Tests Files (Positive tests), 10.20 Integration Tests Files (Negative tests), and 10.21 Future Work Cases.

7.2 Test Suites

7.2.1 testall.sh

```
#!/bin/sh

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

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

# Path to the LLVM compiler
LLC="llc"

# Path to the C compiler
CC="cc"

# Path to the microc compiler.  Usually "./microc.native"
# Try "_build/microc.native" if ocamlbuild was unable to create a symbolic link.
MATCAT="./matcat.native"
#MICROC="_build/microc.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] [.mc 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/.mc//'`
   reffile=`echo $1 | sed 's/.mc$//'`
   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
```

```
${basename}.out" &&
   Run "$MATCAT" "$1" ">" "${basename}.ll" &&
   Run "$LLC" "-relocation-model=pic" "${basename}.ll" ">" "${basename}.s" &&
   Run "$CC" "-o" "${basename}.exe" "${basename}.s" "matrixLibrary.o" "-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/.mc//'`
   reffile=`echo $1 | sed 's/.mc$//'`
   basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

   echo -n "$basename..."

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

   generatedfiles=""

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

   # Report the status and clean up the generated files

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

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

shift `expr $OPTIND - 1`

LLIFail() {
 echo "Could not find the LLVM interpreter \"$LLI\"."
 echo "Check your LLVM installation and/or modify the LLI variable in testall.sh"
 exit 1
}

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


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

7.2.2 run.sh

```
#!/bin/sh
```

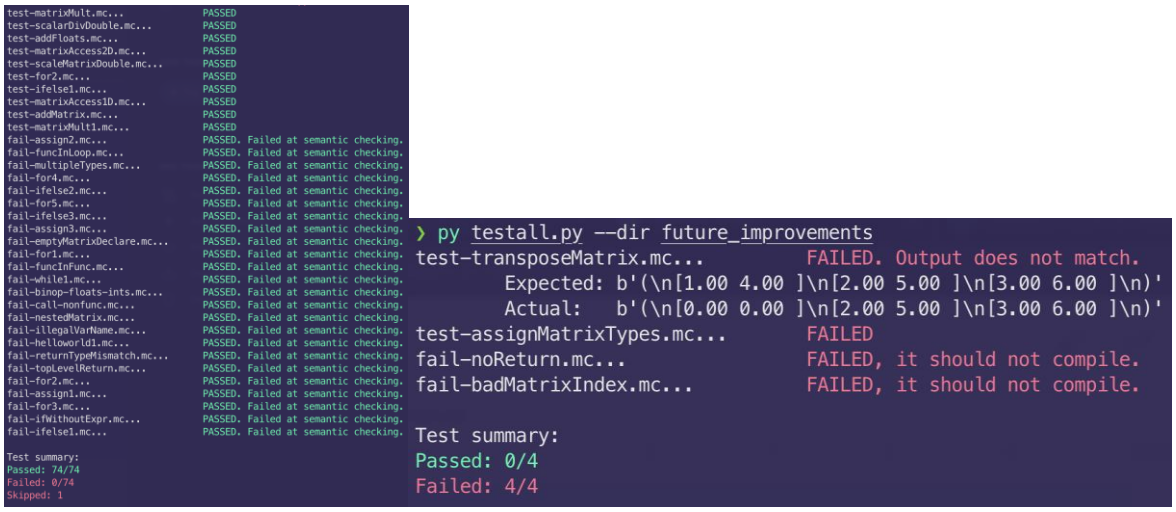
```
# Author:       Andreas
# Usage:        ./mc.sh <filename>
# Description:  This shell script compiles and runs a single .mc file to test a single test
case faster
#               It doesn't check. It simply runs.
LLI="lli"
LLC="llc"
CC="cc"
MATCAT="./matcat.native"
ulimit -t 5
SignalError() {
   if [ $error -eq 0 ]; then
       echo "FAILED"
       error=1
   fi
   echo "  $1"
}
Run() {
   echo $* 1>&2
   eval $* || {
       SignalError "$1 failed on $*"
       return 1
   }
}
basename=`echo $1 | sed 's/.*\\///
                         s/.mc//'`
reffile=`echo $1 | sed 's/.mc$//'`
basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

Run $MATCAT $1 >$basename.ll &&
Run "$LLC" "-relocation-model=pic" "${basename}.ll" &&
Run "$CC" "-o" "${basename}.exe" "${basename}.s" "matrixLibrary.o" "-lm"&&
Run ./$basename.exe

rm -f "$basename.exe"
rm -f "$basename.s"
rm -f "$basename.ll"
```

7.3 Automation

Our test cases are run via shell scripts. We have createTest.sh which auto generates .mc and .out/.err files in tests/ with a given name, testall.sh which runs diff on these corresponding files, and run.sh which runs a single .mc file. Overall, we found it helpful to run our test cases both by comparing output to .out/.err, and by checking for error after every compilation step with the $? command.

Other than the test suite that comes with MicroC, we also implemented another test script with bookkeeping, pretty-printing and directory selecting features.

Some screenshots:



7.4 Division of Labour

Davit and Mariam authored positive test cases for matrix library features as they were developed. James wrote both positive and negative test cases, primarily concerned with finding bugs and edge cases especially with the new matrix type. Andreas wrote testall.py which runs cases with prettier printing, and contributed to tests/ along the way.

8 Lessons Learned and Advice

8.1 Mariam Khmaladze

Working on the project did not always go with the flow, we had challenges but it was nice to solve them. After Davit implemented matrix type, it gave us a big push. It was interesting to work on semant, codegen, all the other files and see how different pieces made the whole feature work. Ocaml is hard to grasp (and even though we did not implement matrix and other features fully in c) I learned quite a bit by coding some of the features. Compilers are hard to write, there are lots of small details to take into account. Debugging takes time and is sometimes nerve wrecking, especially when you write in Ocaml, but they are manageable and it is really important to have teammates who are willing to work on them with you.

8.2 Davit Barblishvili

Working on the semester-long project with other team members while studying a functional programming language is not an easy task. First of all, communication among team members is the most crucial aspect of completing the project. Here are the reasons I think collaboration is needed to complete the project. When you start studying Ocaml, LLVM, and other important aspects of writing a compiler, nothing makes sense. Everything seems chaotic and it is not easy to connect all the dots. Once you understand what all the steps are to finish the project, there is not enough time, and that is where the importance of teamwork comes.

Second, understand Ocaml and when I say this I mean understand what each character of Ocaml does. If I were to start this project again, I would spend the first month to completely understand Ocaml and the last two months to work on the project because you may have a great idea for your little language, but not understanding Ocaml fully might prevent you from implementing it.

Lastly, I would like to emphasize what I have gained from this project: 1) learned functional programming and touched Lambda calculus (the most interesting topic in the class); 2) Understood why NFAs and DFAs are important; 3) mastered regular expressions; and 4) most importantly, I have a better understanding of the other programming languages such as Java, C/C++, Python.

P.S If your assigned TA is not supportive, do not wait to mention this to the instructional team. You would need guidance/support from TA to successfully complete the project.

8.3 James Ryan

Despite being a Senior, this was the first semester long group project I have taken part in (UI Design doesn’t count) and thus have learned plenty about cooperative workflows, communication, division of labor. Software like Slack and When2Meet were crucial for increasing transparency and getting in touch with each other fairly quickly; with one team

member in another time zone, this was especially important. Also, since Andreas joined our group later in the semester, and brought along with him so many workflow optimizations - GoogleDocs document outline, multiple Slack workstreams, handy VSCode extensions - it is striking to see the boost in productivity from this point on.

It was interesting to serve as the Tester and always think in terms of edge cases and worst case scenarios; if there is an uncaught bug it's very much on me. Although it was tempting to exhaustively test all of the functionality we planned on implementing early, the better path was to write test cases incrementally as we developed. This is because it is impossible to predict how your language will evolve as you learn more - we scrapped the Vector data type a week before finals. Finally as advice, it is very helpful to develop a good rapport with a TA. Our assigned TA, the one we discussed our language with for plenty of the semester, was not the most responsive and at times gave strange advice. When we started attending other office hours and meeting regularly with Xijiao, we were able to work towards a much clearer goal.

8.4 Andreas Cheng

Building a compiler from a parser, scanner, semantic checker and assembly code generator is really amazing (but tedious in learning OCaml and LLVM at the same time). The development process is not smooth but overcoming different challenges in the project makes me realize how difficult it is to write a usable language. These experiences help me to imagine how other common languages work behind the scene.

Other than learning how to build a working compiler, I also learned how to code in OCaml, the first functional programming language I have ever used. Although the language supports that I found in Visual Studio Code are quite incomparable to other popular languages, coding in OCaml changes the way I think in coding.

Like many other pieces of advice, "start as soon as possible." The following are some solid "getting-started" advice I can think of: Although it may take quite a while until you get all the concepts in Ocaml and LLVMs, starting with learning some Git workflows or being familiar with the coding environment at the very beginning would really help from the mid-stage to the final stage of the project. Also, try to think of some protocols with your team to share knowledge in coding, git, IDE, or anything related to the project. This would really help in making you a better team.

One last tip: try not to write the final report with Google Doc. It is such a pain to paste the code in the appendix.

9 Acknowledgements

Special thanks to Xijiao Li, our TA advisor. Thanks to Professor Edwards for providing MicroC. We referenced the following past projects:

1. MicroC - We start our project with MicroC. The test-suites also originated from MicroC.
2. PixelPlusPlus - Our Vdecl took a lot of their ideas in declaring variables as a statement: make sure the symbol table is updated in each statement and the way we look for which scope a variable belongs to.
3. Shoo - We followed their format and the designs in their LRM and Final Report. We also referenced their project files
4. Coral - We followed their format and the designs in their LRM and Final Report. We also referenced their project files

10 Appendix

10.1 Test Cases with Color

The test cases with plain formatting and signatures can be found in the Appendix in: 10.19 Integration Tests Files (Positive tests), 10.20 Integration Tests Files (Negative tests), and 10.21 Future Work Cases. The cases below are with syntax highlighting.

10.1.1 Positive Test Cases

10.1.1.1 test-transpose.mc

```
func main() int
{
    matrix a;
    matrix c;
    a = [[1,2,3],[4,5,6]];
    c = transpose(a);

    printm(c);
    return 0;

}
```

10.1.1.2 test-transpose.out

```
(
[1.00 4.00 ]
[2.00 5.00 ]
[3.00 6.00 ]
)
```

10.1.1.3 test-addFloats.mc

```
func main() int {
    double a;
    double b;
    double c;

    a = 5.1;
    b = 7.2;
    c = a + b;
    printd(c);

    return 0;
}
```

10.1.1.4 test-addFloats.out

```
12.3
```

10.1.1.5 test-addInts.mc

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

    a = 5;
    b = 7;
    c = a + b;
    print(c);

    return 0;
}
```

10.1.1.6 test-addInts.out

```
12
```

10.1.1.7 test-addMatrix.mc

```
(
[4.00 4.00 4.00 ]
[5.00 8.00 11.00 ]
[2.00 4.00 6.00 ]
)
```

10.1.1.8 test-addMatrix.out

```
(
[4.00 4.00 4.00 ]
[5.00 8.00 11.00 ]
[2.00 4.00 6.00 ]
)
```

10.1.1.9 test-boolAssign.mc

```
func main() int
{

    bool b = false;

    printb(b);

    return 0;
}
```

10.1.1.10 test-boolAssign.out

```
0
```

10.1.1.11 test-comparison.mc

```
func main() int
{

    bool b1;
    bool b2;
    bool r;
    int a;
    int b;
    bool geq;
    bool neq;
    bool lt;


    b1 = false;
    b2 = true;
    neq = b1 != b2;
    printb(neq); // 1
    a = 1;
    b = 0;
    geq = a >= b;
    lt = a < b;

    printb(geq); // 1
    printb(lt); // 0

    return 0;
}
```

10.1.1.12 test-comparison.out

```
1
1
0
```

10.1.1.13 test-detMatrix.mc

```
func main() int {
    matrix a;
    double d;
    a =
[[1,0,1,1,2],[2,1,0,1,2],[3,1,2,1,2],[1,2,1,3,2],[1,4,5,2,2]];
    d = det(a);

    printd(d);

    return 0;
}
```

10.1.1.14 test-detMatrix.out

```
52
```

10.1.1.15 test-detMatrix1.mc

```
func main() int {
    matrix a;
    double d;
    a = [[1,2],[5,2]];
    d = det(a);

    printd(d);

    return 0;
}
```

10.1.1.16 test-detMatrix1.out

```
-8
```

10.1.1.17 test-dotProduct.mc

```
func main() int{
    matrix a;
    matrix b;
```

```
    double c;

    a = [[1,2,3]];
    b = [[2,3,4]];

    c = a dot b;
    printd(c);


    return 0;
}
```

10.1.1.18 test-dotProduct.out

```
20
```

10.1.1.19 test-dotProduct1.mc

```
func main() int{
    matrix a;
    matrix b;
    double c;

    a = [[1,2],[3,2]];
    b = [[2,3],[2,1]];

    c = a dot b;
    printd(c);


    return 0;
}
```

10.1.1.20 test-dotProduct1.out

```
16
```

10.1.1.21 test-for1.mc

```
func main() int
```

```
{
  int i;

  for (i = 0; i < 1 ; i = i + 1) {
      printStr("Loop Success");
  }

  return 0;
}
```

10.1.1.22 test-for1.out

```
Loop Success
```

10.1.1.23 test-for2.mc

```
func main() int
{
  int i;
  int j;
  for (i = 0; i < 1 ; i = i + 1) {
    for (j = 0; j <= i ; j = j + 1) { // declare i inside loop
      printStr("nested loop");

    }
  }

  return 0;
}
```

10.1.1.24 test-for2.out

```
nested loop
```

10.1.1.25 test-ifelse1.mc

```
func main() int
{

  int f = 2;
```

```
  if(f > 0) {
      printStr("greater");
  } else {
      printStr("lesser");
  }

  return 0;
}
```

10.1.1.26 test-ifelse1.out

```
greater
```

10.1.1.27 test-ifelse2.mc

```
func main() int
{

  bool b;
  b = true;


  if(b) {
      printStr("goodprint");
  }

  if(!b) {
      printStr("badprint");
  }


  return 0;
}
```

10.1.1.28 test-ifelse2.out

```
goodprint
```

10.1.1.29 test-ifelse3.mc

```
func main() int
{

  bool b = false;

  if(b) {
      printStr("badprint1");
  } else if(true) {
      printStr("goodprint");
  } else {
      printStr("badprint2");
  }

  return 0;
}
```

10.1.1.30 test-ifelse3.out

```
goodprint
```

10.1.1.31 test-inverseMatrix.mc

```
func main() int
{
    matrix a;
    matrix c;
    a = [[1,2],[4,5]];
    c = inv(a);

    printm(c);

    return 0;

}
```

10.1.1.32 test-inverseMatrix.out

```
(
```

```
[-1.67 0.67 ]
[1.33 -0.33 ]
)
```

10.1.1.33 test-inverseMatrix1.mc

```
func main() int
{
    matrix a;
    matrix c;
    a = [[2,5],[5,1]];
    c = inv(a);

    printm(c);

    return 0;

}
```

10.1.1.34 test-inverseMatrix1.out

```
(
[-0.04 0.22 ]
[0.22 -0.09 ]
)
```

10.1.1.35 test-inverseMatrix2.mc

```
func main() int
{
    matrix a;
    matrix c;
    a = [[2,5,1],[5,1,6],[3,1,2]];
    c = inv(a);

    printm(c);

    return 0;

}
```

10.1.1.36 test-inverseMatrix2.out

```
(
[-0.12 -0.26 0.85 ]
[0.24 0.03 -0.21 ]
[0.06 0.38 -0.68 ]
)
```

10.1.1.37 test-isInvertible.mc

```
func main() int {
    matrix a;
    a = [[3,3],[5,5]];
    isInv(a);

    return 0;
}
```

10.1.1.38 test-isInvertible.out

```
This matrix is not invertible
```

10.1.1.39 test-isInvertible1.mc

```
func main() int {
    matrix a;
    a = [[5,6],[8,8]];
    isInv(a);

    return 0;
}
```

10.1.1.40 test-isInvertible1.out

```
This matrix is invertible
```

10.1.1.41 test-manyPrints.mc

```
func main() int
{
    printStr("hello world");
```

```
    printStr("with more words");
    return 0;

}
```

10.1.1.42 test-manyPrints.out

```
hello world
with more words
```

10.1.1.43 test-matrix-function1.mc

```
func m() matrix {
        return [[3, 4]];
}

func main() int {
    matrix m;
    m = m();
    printm(m);

    return 0;
}
```

10.1.1.44 test-matrix-function1.out

```
(
[3.00 4.00 ]
)
```

10.1.1.45 test-matrixAccess1D.mc

```
func main() int{
    matrix a;
    a = [[1,2,3],[4,5,6]];
    printm(a[1,:]);

    return 0;
}
```

10.1.1.46 test-matrixAccess1D.out

```
(
[4.00 5.00 6.00 ]
)
```

10.1.1.47 test-matrixAccess2D.mc

```
func main() int {
    matrix c;
    c = [[1,2],[3,4]];
    printm(c[0][0]);

    return 0;
}
```

10.1.1.48 test-matrixAccess2D.out

```
(
[1.00 ]
)
```

10.1.1.49 test-matrixAccessCol.mc

```
func main() int{
    matrix a;
    a = [[1,2,3],[4,5,6],[10,20,34],[3,7,6]];
    printm(a[:,1]);

    return 0;
}
```

10.1.1.50 test-matrixAccessCol.out

```
(
[2.00 5.00 20.00 7.00 ]
)
```

10.1.1.51 test-matrixAccessDiagonal.mc

```
func main() int {
```

```
    matrix a;
    matrix b;

    a = [[1,2,3],[4,5,6],[7,8,9]];
    b = a[:,:];

    printm(b);


    return 0;
}
```

10.1.1.52 test-matrixAccessDiagonal.out

```
(
[1.00 5.00 9.00 ]
)
```

10.1.1.53 test-matrixAdd1.mc

```
func main() int {
    matrix c;
    c = [[1,2],[3,4]] + [[3,2],[5,4]];
    printm(c);

    return 0;
}
```

10.1.1.54 test-matrixAdd1.out

```
(
[4.00 4.00 ]
[8.00 8.00 ]
)
```

10.1.1.55 test-matrixMult.mc

```
func main() int {
    matrix a;
    matrix b;
    matrix c;
```

```
    a = [[1,2,3],[4,5,6],[7,8,9]];
    b = [[10,11,12],[13,14,15],[16,17,18]];

    c = a * b;
    printm(c);

    return 0;
}
```

10.1.1.56 test-matrixMult.out

```
(
[84.00 90.00 96.00 ]
[201.00 216.00 231.00 ]
[318.00 342.00 366.00 ]
)
```

10.1.1.57 test-matrixMult1.mc

```
func main() int {
    matrix a;
    matrix c;

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


    c = a * a;
    printm(c);

    return 0;
}
```

10.1.1.58 test-matrixMult1.out

```
(
[9.00 12.00 ]
[24.00 33.00 ]
)
```

10.1.1.59 test-matrixPower.mc

```
func main() int {
    matrix a;
    matrix b;
    a = [[1,2,3],[4,5,6],[7,8,9]];
    b = a^5;

    printm(b);

    return 0;

}
```

10.1.1.60 test-matrixPower.out

```
(
[121824.00 149688.00 177552.00 ]
[275886.00 338985.00 402084.00 ]
[429948.00 528282.00 626616.00 ]
)
```

10.1.1.61 test-matrixPower1.mc

```
func main() int {
    matrix a;
    matrix b;
    a = [[1,2,3],[4,5,6],[7,8,9]];
    b = a^0;

    printm(b);

    return 0;

}
```

10.1.1.62 test-matrixPower1.out

```
(
[1.00 0.00 0.00 ]
[0.00 1.00 0.00 ]
[0.00 0.00 1.00 ]
```

```
)
```

10.1.1.63 test-matrixPower2.mc

```
func main() int {
    matrix a;
    matrix b;
    a = [[1,2,2],[4,5,6],[7,0,9]];
    b = a^-3;

    printm(b);

    return 0;

}
```

10.1.1.64 test-matrixPower2.out

```
(
[-35.18 13.25 -0.99 ]
[-3.40 1.29 -0.11 ]
[26.45 -9.97 0.75 ]
)
```

10.1.1.65 test-matrixSub.mc

```
func main() int {
    matrix a;
    matrix b;
    matrix c;

    a = [[1,2,3],[3,4,5],[1,2,3]];
    b =[[3,2,1],[2,4,6],[1,2,3]];
    c = a - b;
    printm(c);
    return 0;
}
```

10.1.1.66 test-matrixSub.out

```
(
[-2.00 0.00 2.00 ]
```

```
[1.00 0.00 -1.00 ]
[0.00 0.00 0.00 ]
)
```

10.1.1.67 test-matrixSymmetry.mc

```
func main() int {
    int test;
    matrix a = [[1,2,3],[2,4,5],[3,5,6]];
    matrix b;

    test = check_symmetry(a);
    if(test == 1){
        printStr("The matrix is symmetric");
        printStr("Here is the original matrix:");
        printm(a);
        printStr("");
        printStr("Here is the transposed matrix:");
        b = transpose(a);
        printm(b);

    }else{
        printStr("The matrix is not symmetric");
    }
}
```

10.1.1.68 test-matrixSymmetry.out

```
The matrix is symmetric
Here is the original matrix:
(
[1.00 2.00 3.00 ]
[2.00 4.00 5.00 ]
[3.00 5.00 6.00 ]
)

Here is the transposed matrix:
(
[1.00 2.00 3.00 ]
[2.00 4.00 5.00 ]
[3.00 5.00 6.00 ]
)
```

10.1.1.69 test-multipleAssignment.mc

```
func main() int {

    int a;
    int b;
    int c;
    a = b = c = 5;
    if(a == b && a == c) {
        printStr("equal");
    }
    else {
        printStr("oh no");
    }


    return 0;
}
```

10.1.1.70 test-multipleAssignment.out

```
equal
```

10.1.1.71 Test-print.mc

```
func main() int
{
    printStr("hello world");
    return 0;

}
```

10.1.1.72 test-print.out

```
hello world
```

10.1.1.73 test-printb.mc

```
func main() int {

```

```
    printb(true);
    return 0;

}
```

10.1.1.74 test-printb.mc

```
1
```

10.1.1.75 test-printMatrix.mc

```
func main() int {
    matrix m;
    m = [[1,2,3],[4,5,6],[10,20,30],[100,255,560]];
    printm(m);

    return 0;
}
```

10.1.1.76 test-printMatrix.out

```
(
[1.00 2.00 3.00 ]
[4.00 5.00 6.00 ]
[10.00 20.00 30.00 ]
[100.00 255.00 560.00 ]
)
```

10.1.1.77 test-printstring.mc

```
func main() int
{
    printStr("hello world");
    return 0;
}
```

10.1.1.78 test-printstring.out

```
hello world
```

10.1.1.79 test-printvoid.mc

```
func printVoid() void {
    printStr("Hello world");
}

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

10.1.1.80 test-printvoid.out

```
Hello world
```

10.1.1.81 test-recursion.mc

```
func foo(int n) int {
    if (n == 1) {
        return 0;
    }
    return bar(n - 1);
}

func bar(int n) int {
    if (n == 1) {
        return 0;
    }
    return foo(n - 1);
}

func main() int {
    print(foo(5));
    return 0;
}
```

10.1.1.82 test-recursion.out

```
0
```

test-rotate90.mc

```
func main() int {
    matrix a = [[1,2,3],[4,5,6],[7,8,9]];
    rotate90(a);
    printm(a);
    return 0;
}
```

10.1.1.83 test-rotate90.out

```
(
[7.00 4.00 1.00 ]
[8.00 5.00 2.00 ]
[9.00 6.00 3.00 ]
)
```

10.1.1.84 test-rowAccessRetMatrix.mc

```
func main() int {
    matrix m;
    m = [[1,2,3]];

    int i = 0;
    while(i < 3) {
        i = i+1;
        m = m[0,:];
        printm(m);
    }


    return 0;
}
```

10.1.1.85 test-rowAccessRetMatrix.out

```
(
[1.00 2.00 3.00 ]
)
(
[1.00 2.00 3.00 ]
)
(
```

```
[1.00 2.00 3.00 ]
)
```

10.1.1.86 test-scalarDivDouble.mc

```
func main() int {
    matrix a;
    matrix b;

    a = [[1,2,3],[4,5,6],[2,10,2]];
    b = a / 2.5;

    printm(b);

    return 0;

}
```

10.1.1.87 test-scalarDivDouble.out

```
(
[0.40 0.80 1.20 ]
[1.60 2.00 2.40 ]
[0.80 4.00 0.80 ]
)
```

10.1.1.88 test-scalarDivMatrix.mc

```
func main() int {
    matrix a;
    matrix b;

    a = [[1,2,3],[4,5,6],[2,10,2]];
    b = a / 2;

    printm(b);

    return 0;

}
```

10.1.1.89 test-scalarDivMatrix.out

```
(
[0.50 1.00 1.50 ]
[2.00 2.50 3.00 ]
[1.00 5.00 1.00 ]
)
```

10.1.1.90 test-scaleMatrix.mc

```
func main() int {
    matrix a;
    matrix b;

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

    printm(b);

    return 0;
}
```

10.1.1.91 test-scaleMatrix.out

```
(
[3.00 6.00 9.00 ]
[12.00 15.00 18.00 ]
)
```

10.1.1.92 test-scaleMatrix1.mc

```
func main() int {
    matrix a;
    matrix b;

    a = [[1,2,3],[4,5,6],[2,10,2]];
    b = 1 * a;

    printm(b);

    return 0;
```

```
}
```

10.1.1.93 test-scaleMatrix1.out

```
(
[1.00 2.00 3.00 ]
[4.00 5.00 6.00 ]
[2.00 10.00 2.00 ]
)
```

10.1.1.94 test-scaleMatrixDouble.mc

```
func main() int {
    matrix a;
    matrix b;

    a = [[1,2,3],[4,5,6],[2,10,2]];
    b = 2.5 * a;

    printm(b);

    return 0;

}
```

10.1.1.95 test-scaleMatrixDouble.out

```
(
[2.50 5.00 7.50 ]
[10.00 12.50 15.00 ]
[5.00 25.00 5.00 ]
)
```

10.1.1.96 test-while1.mc

```
func main() int
{

    bool b = false;

    while(b) {}
```

```
    printStr("pass");

    return 0;
}
```

10.1.1.97 test-while1.out

```
pass
```

10.1.1.98 test-while2..mc

```
func main() int
{

    while(2 != (2 * 1)) {}

    printStr("pass");

    return 0;
}
```

10.1.1.99 test-while2.out

```
pass
```

10.1.2 Negative Test Cases

10.1.2.1 fail-assign1.err

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

10.1.2.2 fail-assign1.mc

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

```
  bool b;

  i = 42;
  i = 10;
  b = true;
  b = false;
  i = false; /* Fail: assigning a bool to an integer */
  return 0;
}
```

10.1.2.3 fail-assign2.err

```
Fatal error: exception Failure("illegal assignment bool = int in b = 48")
```

10.1.2.4 fail-assign2.mc

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

  b = 48; /* Fail: assigning an integer to a bool */
  return 0;
}
```

10.1.2.5 fail-assign3.err

```
Fatal error: exception Failure("illegal assignment int = void in i = myvoid()")
```

10.1.2.6 fail-assign3.mc

```
func myvoid() void
{
  return;
}

func main() int
{
  int i;

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

```
}
```

10.1.2.7 fail-binop-floats-ints.err

```
Fatal error: exception Failure("illegal binary operator int + double in i + a")
```

10.1.2.8 fail-binop-floats-ints.mc

```
func main() int {

    int i;
    double a;
    double b;
    double c;

    i = 5;
    a = 6.0;
    c = i + a;

    printd(c);

    return 0;
}
```

10.1.2.9 fail-call-nofunc.err

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

10.1.2.10 fail-call-nofunc.mc

```
func main() int {

    int foo;
    foo = 5;
    foo();

    return 0;
```

```
}
```

10.1.2.11 fail-emptyMatrixDeclare.err

```
Fatal error: exception Failure("hd")
```

10.1.2.12 fail-emptyMatrixDeclare.mc

```
func main() int {
    matrix m;
    m = [[], [], []];
    printm(m);

    return 0;
}
```

10.1.2.13 fail-for1.err

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

10.1.2.14 fail-for1.mc

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

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

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

  return 0;
}
```

10.1.2.15 fail-for2.err

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

10.1.2.16 fail-for2.mc

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

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

  return 0;
}
```

10.1.2.17 fail-for3.err

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

10.1.2.18 fail-for3.mc

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

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

  return 0;
}
```

10.1.2.19 fail-for4.err

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

10.1.2.20 fail-for4.mc

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

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

  return 0;
}
```

10.1.2.21 fail-for5.err

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

10.1.2.22 fail-for5.mc

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

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

  return 0;
}
```

10.1.2.23 fail-funcInFunc.err

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

10.1.2.24 fail-funcInFunc.mc

```
func main() int {



    func foo() int {
        return 0;
    }



    return 0;
```

```
}
```

10.1.2.25 fail-funcInLoop.err

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

10.1.2.26 fail-funcInLoop.mc

```
func main() int {

    while(true) {

        func foo() int {
            return 0;
        }

    }

    return 0;
}
```

10.1.2.27 fail-helloworld1.err

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

10.1.2.28 fail-helloworld1.mc

```
func main() int
{
    printStr("hell) // malformed
    return 0;
}
```

10.1.2.29 fail-ifelse1.err

```
Fatal error: exception Failure("expected Boolean expression in 5")
```

10.1.2.30 fail-ifelse2.mc

```
func main() int
{

  bool b = true;

  if(5) { // non bool
      printStr("goodprint");
  }

  if(!b) {
      printStr("badprint");
  }

  return 0;
}
```

10.1.2.31 fail-ifelse2.err

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

10.1.2.32 fail-ifelse2.mc

```
func main() int
{

  bool b;
  b = true;

  if(b) {
      printStr("hm");
  } else {
      printStr("end?");
  } else if(false) { // elif after else
      printStr("blah");
  }

  return 0;
}
```

10.1.2.33 fail-ifelse3.err

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

10.1.2.34 fail-ifelse3.mc

```
func main() int
{

  bool b;
  b = true;

  if(b) {
      if(!b) {
          if(true) {
              else { // nested but no corresponding if
                  printStr("very nested")
              }
          }
      }
  }

  return 0;
}
```

10.1.2.35 fail-ifWithoutExpr.err

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

10.1.2.36 fail-ifWithoutExpr.mc

```
func main() int {

    if () {
        printStr("no");
    }

    return 0;
}
```

10.1.2.37 fail-illegalVarName.err

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

10.1.2.38 fail-illegalVarName.mc

```
func main() int {

    int $a;
    int 1a;
    1a = 2;

    return 0;
}
```

10.1.2.39 fail-multipleTypes.err

```
Fatal error: exception Failure("duplicate local a")
```

10.1.2.40 fail-multipleTypes.mc

```
func main() int {

    int a;
    double a;

    a = 5;
    a = 7.2;

    return 0;

}
```

10.1.2.41 fail-nestedMatrix.err

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

10.1.2.42 fail-nestedMatrix.mc

```
func main() int {
    matrix m;
    m = [[[1,2,3], [4, 5, 6]], [7, 8, 9]];
    printm(m);

    return 0;
}
```

10.1.2.43 fail-returnTypeMismatch.err

```
Fatal error: exception Failure("return gives double expected int in 5.0")
```

10.1.2.44 fail-returnTypeMismatch.mc

```
func main() int {

    int a;
    a = 1;

    return 5.0;
}
```

10.1.2.45 fail-topLevelReturn.err

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

10.1.2.46 fail-toplevelReturn.mc

```
return;

func main() int {

    return 0;
}
```

10.1.2.47 fail-while1.err

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

10.1.2.48 fail-while1.mc

```
func main() int
{

    int i;
    i = 0;
    while(i) { } // non bool

    return 0;
}
```

10.1.3 Future Work Cases

These are bugs which we were not able to sort out before the deadline, but we are nonetheless aware of.

10.1.3.1 test-redeclareFuncParam.mc

Should be illegal to redeclare the function argument x in the same namespace.

```
func foo(int x) int {
    int x;
    x = 2;
    return x;
}


func main() int {

    print(foo(5));

    return 0;
}
```

10.1.3.2 test-redeclareFuncParam.out

```
2
```

10.1.3.3 test-badMatrixIndex.err

Not caught before runtime. Perhaps cannot be improved upon but distinct from other fail cases.

```
Matrix index out of bound: Success
```

10.1.3.4 test-badMatrixIndex.mc

```
func main() int {
    matrix a;
    a = [[1,2,3]];
    printm(a[1,:]); // runtime error, will not work with diff

    return 0;
}
```

10.1.3.5 test-assignMatrixTypes.mc

Once m is initialized with an integer, reassignment to a matrix of type double is not supported.

```
func main() int {
    matrix m;
    int a = 1;
    double b = 44.25;

    m = [[a]];
    m = [[b]]; // fails here
    m = [[a], [b]];
    printStr("OK");

    return 0;
}
```

10.1.3.6 test-assignMatrixTypes.out

```
N/A
```

10.2 Project Log

```
commit 56f2399b0c7eb199ca1c87d4315714459ea97fdd
Author: Andreas <and@reas.me>
Date:   Tue Apr 27 08:24:11 2021 +0800

    fix Makefile tar ball issue

commit 81f8ae02e53150a62feb21175c3a76d246865c9c
Author: Andreas <and@reas.me>
Date:   Tue Apr 27 08:12:32 2021 +0800

    refine Makefile

commit 3aa417767f73c92f3abfd9e6d629eab7b8456d8c
Author: Andreas <and@reas.me>
Date:   Tue Apr 27 05:28:43 2021 +0800

    sign the _tags as well

commit 0953dfc4acbaf07a58f2d2ac339123d265aee5ef
Author: Andreas <and@reas.me>
Date:   Tue Apr 27 05:20:51 2021 +0800

    refine README

commit cecff481169cd80919e8eeaa73cc1494da22b2d4
Author: Andreas <and@reas.me>
Date:   Tue Apr 27 05:18:57 2021 +0800

    refine Makefile and various files

    also signed the Makefile

commit 1693eacd7a43d9facc5cc2000930df67c7d33810
Author: Andreas <and@reas.me>
Date:   Tue Apr 27 04:56:46 2021 +0800

    refine .gitignore, .merlin, and remove mc.sh

    mc.sh: dup with run.sh

commit 85785c498bf04bbcd6c9a70f97f9e18940fe7026
Author: Andreas <and@reas.me>
Date:   Tue Apr 27 04:55:22 2021 +0800

    sign all codes for everyone

commit b5d18ce029bc09033093db4a322c0e1ee925e283
Author: Andreas <and@reas.me>
Date:   Tue Apr 27 04:27:10 2021 +0800

    refine style of autosign

commit 496d7b2bb234df420a902456cad38bee050ae422
Author: Andreas <and@reas.me>
Date:   Tue Apr 27 03:53:11 2021 +0800

    sign the demo and fix grammar (author[s])

commit 12cf59effe1fe10064904fcaf689945729f24382
Author: Andreas <and@reas.me>
```

```
Date:   Tue Apr 27 03:48:33 2021 +0800

    sign testcases in future_improvements

commit 9081c3425c5e58d19a6f0bb8daa0c9c014510249
Author: Andreas <and@reas.me>
Date:   Tue Apr 27 03:45:37 2021 +0800

    Sign testcases with 2 authors

commit 4d04dd00314e009fc36eec918e76228101f34898
Author: Andreas <and@reas.me>
Date:   Tue Apr 27 03:42:04 2021 +0800

    sign testcases with single author

    Note: the script now ignores MicroC creation/removal commits:
    3b6f80de
    0e29cf4

commit 9cde820ff2fa51b01fb900f38b055023de008635
Author: Andreas <and@reas.me>
Date:   Tue Apr 27 03:36:04 2021 +0800

    autosign files with one author

commit 73111e008a35fd514349c8a8ded35b3f18840edf
Author: Andreas <and@reas.me>
Date:   Tue Apr 27 03:33:07 2021 +0800

    remove overview.txt

commit f29760476280178b1c133ae9bfe4f84745354e0b
Author: Andreas <and@reas.me>
Date:   Mon Apr 26 23:33:19 2021 +0800

    update readme; add dot product testcase

commit 8736093c89c8af0f001b57e22358c6e635b65978
Merge: 0627c34 295593c
Author: Andreas <and@reas.me>
Date:   Mon Apr 26 21:34:21 2021 +0800

    Merge branch 'main' of github.com:davitbarblishvili/Matcat into main

commit 0627c34b59a958078216198ee8c0fb714b3ba9dd
Author: Andreas <and@reas.me>
Date:   Mon Apr 26 21:34:02 2021 +0800

    Add colorama in Dockerfile and README

commit 295593c79419504aab5094cf6730b352191526b0
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Mon Apr 26 09:32:52 2021 -0400

    added demo dir.

commit c79f7c76ecf5e30d4fe65a83d49f0b56787001d8
Author: Andreas <and@reas.me>
Date:   Mon Apr 26 21:08:07 2021 +0800

    Update Dockerfile and README Docker instruction
```

```
    Note:
    - repository name must be lowercase"
    - mcDocker" is wrong

commit 7dd534520fdf8c0856776dbca209fee0da4923c6
Author: Andreas <and@reas.me>
Date:   Mon Apr 26 20:22:14 2021 +0800

    refine demension checking in matrxAdd and matrxSub

commit 27af24a6c03fdd28e6cbbf38ea58468f750a5f30
Author: Andreas <and@reas.me>
Date:   Mon Apr 26 18:40:39 2021 +0800

    change order of fdecl as well

commit 507942397817b737fef818a83f133aa95db0b5c7
Author: Andreas <and@reas.me>
Date:   Mon Apr 26 18:39:26 2021 +0800

    1st step in getting rid of main: ast working...

commit 9c34b5bf57a209acaace87130dd629f3ed4a7eaf
Author: Andreas <and@reas.me>
Date:   Mon Apr 26 18:01:43 2021 +0800

    Update README

    - add URL to some texts
    - refine headers
    - refine the grammar

commit 927c1956c186bef3340909e8fc74089e4b8b443e
Author: Andreas <and@reas.me>
Date:   Mon Apr 26 17:51:24 2021 +0800

    update README and Makefile; clean up

    README:
    - Add Test cases for future improvements
    - grammar fixes

    Makefile: clean when make by default

commit 25687ea2bb45bf8070554d7c5bd0a36a52decd42
Author: Andreas <and@reas.me>
Date:   Mon Apr 26 17:30:22 2021 +0800

    refine testall.py to take test_dir as argument

    This is to test the future improvements
    The command:
    py testall.py --dir future_improvements

commit 06f22e8c63eb003195235ff4edd2300eb3675b8a
Author: Andreas <and@reas.me>
Date:   Mon Apr 26 17:30:03 2021 +0800

    move some testcases to future improvements

commit 388c1d0de09ee0c57e3c580f2de20a5d8460b2ab
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Apr 25 22:20:43 2021 -0400
```

```
    updated diagonal access to [:,;]

commit 15c4f763c6d3eb276c162d8f26a52ea1e06a7371
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Apr 25 20:06:49 2021 -0400

    clean up && some error fixing

commit d8d78dd188be78443f9bd090451980be4f866512
Author: Andreas <and@reas.me>
Date:   Mon Apr 26 03:48:07 2021 +0800

    fix return problem

commit b42f635f2adc3a1cbf2ebcff40584a49970597d5
Merge: c347052 4ff2b55 5696a6a
Author: Andreas <and@reas.me>
Date:   Mon Apr 26 03:07:51 2021 +0800

    On main: bs

commit 4ff2b55801aa8a65eca93875cf51c42648a4ade2
Author: Andreas <and@reas.me>
Date:   Mon Apr 26 03:07:51 2021 +0800

    index on main: c347052 beautify parser; ban global initialization

commit 5696a6a87b0c4a0a3d3a26b8a67c541f3e01b760
Author: Andreas <and@reas.me>
Date:   Mon Apr 26 03:07:51 2021 +0800

    untracked files on main: c347052 beautify parser; ban global initialization

commit c3470525ab5361355666544757a941d7988b69d0
Author: Andreas <and@reas.me>
Date:   Mon Apr 26 01:55:15 2021 +0800

    beautify parser; ban global initialization

commit 9cefd5549d3303278b9d4f81654080a6601fc008
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Apr 25 10:50:05 2021 -0400

    clean up

commit 5f8bfabbe7389ef828185dbd692a3a7f9e2468ca
Author: James Ryan <jamesryan@Jamess-MBP-2.fios-router.home>
Date:   Sun Apr 25 10:26:05 2021 -0400

    more tests

commit a5ea91f49803df1420354a9721798f1380f8785c
Author: Andreas <and@reas.me>
Date:   Sun Apr 25 20:56:30 2021 +0800

    Refine styles and update README.md

    Remove/add some newlines in Makefile
    Add references and new test-suite description in README.md

commit ad2a2e5d09224f7d08367208ed689b4720050d21
Author: Andreas <and@reas.me>
```

```
Date:   Sun Apr 25 20:44:27 2021 +0800

    refine exprimental testall; edit manyPrints.out

    removed whitespace in manyPrints.out to suit the stricter testall.py

commit 946b5b6fb40d4de1406a3e99631aed8cac36ef18
Author: Andreas <and@reas.me>
Date:   Sun Apr 25 19:17:37 2021 +0800

    add immediate clean up in testall.py

commit 8d21762979ce1901d9549f29490e2c390e488ff2
Author: Andreas <and@reas.me>
Date:   Sun Apr 25 19:12:49 2021 +0800

    add basic of fancy testall.py

commit c811fe50f84ac43fc8d6f6d74dcaa72b557373a6
Author: Andreas <and@reas.me>
Date:   Sun Apr 25 19:12:28 2021 +0800

    refine run.sh

    Note:
    That ">" "${basename}.s" part is not required...

commit eee48e22a04688cd3ef4550601566cd598e89513
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Apr 25 01:06:49 2021 -0400

    added the function that checks if matrix is symmetric or not

    A = A^T

commit b163457ad1a53a8c3c574e5d4f859d695cfd06c7
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Apr 25 00:43:37 2021 -0400

    added matrix rotation by 90

commit 3d9a388a051335ac86019e5817f1c76722f56037
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sat Apr 24 23:06:45 2021 -0400

    fixed shift/reduce conflicts

commit 88b9e90caeb6ed5c140968b7fa4df9dcb4cb9118
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sat Apr 24 21:28:13 2021 -0400

    Added matrix exponentiation

    1) matrix^i  --> returns ith power of the matrix.
    2) matrix^0 --> returns identity matrix of the dimension of the input matrix
    3) matrix^-i --> returns ith power of matrix^-1 (inverse of the matrix).

commit 0c6fa90c2c0aa5a6354a89dba78bf3a0135973a5
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sat Apr 24 20:49:52 2021 -0400

    updated matrix structure in semant and codegen
```

```
    Nothing has changed that affected already existed functions.

commit 17b6f4adcdfecc6c31f2551a77bb7c66057ecc48
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sat Apr 24 19:10:22 2021 -0400

    clean up and fix some errors/warnings

commit 782ee25590063cb2083bd0bd2e3dc54a0b422a50
Merge: 1edf827 891030c
Author: Andreas <45086238+andHW@users.noreply.github.com>
Date:   Sun Apr 25 05:10:19 2021 +0800

    Merge pull request #1 from davitbarblishvili/Vdecl

    Vdecl merge

    Variable declarations are now statements.
    symbol_table and symbol are everywhere in semant.ml and codegen.ml
    The matrix is also working.

    Our big reference:
    https://github.com/maobowen/PixelPlusPlus

commit 891030ccdb98ba503f8b7e3b28b9db543cd0ea6f
Author: Andreas <and@reas.me>
Date:   Sun Apr 25 04:49:20 2021 +0800

    fix noassign problem that caused matrix failed

commit ef91e8ce90886a30146996a16e35a1a58c2e6266
Author: Andreas <and@reas.me>
Date:   Sun Apr 25 04:22:02 2021 +0800

    fixed semant for matrix

commit e2ba42df982f3aaf17b9e3fd4829bf2af4d53a3c
Author: Andreas <and@reas.me>
Date:   Sun Apr 25 03:53:54 2021 +0800

    minor fix in sast

commit 45e74ed3b9d12c6ba156b29409c5bb0950f6e0f4
Author: Andreas <and@reas.me>
Date:   Sat Apr 24 22:14:36 2021 +0800

    update working codegen @_@

    TODO: figure how that matrix part works with new symbol table stuff.

commit ba3753edc5cb8904da0857d13b33304eed0b5c39
Author: Andreas <and@reas.me>
Date:   Sat Apr 24 20:59:16 2021 +0800

    Ast and Sast are working.

    TODO:
    - finish codegen
    - figure out how expressions in matrix works

commit 5ad88f12bdd7791e57273102432e540ed2b8bdb8
Author: Andreas <and@reas.me>
Date:   Sat Apr 24 20:37:38 2021 +0800
```

```
    adding vdecl (2nd attempt): working ast

    Testing command:
    make clean && make && ./matcat.native -a temp.mc

commit 1edf8271229c9770ffd035502d14ea88575d4d26
Author: Andreas <and@reas.me>
Date:   Sat Apr 24 20:30:26 2021 +0800

    misc. updates

    - Refine Makefile
    - Refine run.sh
    - Refine README

commit defb69630066998c34bf07d42dbd52704555d638
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sat Apr 24 01:24:02 2021 -0400

    adding matrix exponentiation - work in prrogress.

commit a4bd4eadc9b182f692ee8b1d592c51975108496c
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Fri Apr 23 22:21:51 2021 -0400

    Added matrix diagonal printing

commit caa7c38765b891d4bd8cb9c7dcc0d0eec1f52402
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Fri Apr 23 15:54:23 2021 -0400

    updated row access from matrix[i] to matrix[i,:]

commit a4730bd38ccf2159afd08927527fe93f7256e2ab
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Fri Apr 23 15:42:36 2021 -0400

    added column access for matrix

commit ac0d089747e8fce141b1be8645d90e180f10869b
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Fri Apr 23 00:09:35 2021 -0400

    added 1D matrix access

commit ed9314db0151b291706367a8dcbeb1dbafdc042c
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Thu Apr 22 22:03:11 2021 -0400

    added 2D access for matrix

commit 13b4b4a1aaa5702337a3e9ce3374899f6335a8fb
Author: Andreas <and@reas.me>
Date:   Thu Apr 22 17:59:11 2021 +0800

    fix typo in README

commit bcc2fe334714aab9a36246cd8444c2530fe88b7f
Author: Andreas <and@reas.me>
Date:   Thu Apr 22 17:55:22 2021 +0800

    clean residue in returning mutiple values
```

```
    - parser: fdecl no longer takes a type_list
    - ast, sast, semant, codegen: fd.data_type is not a list anymore

commit ead38d0c04ab82692a83403495da9388599006e8
Author: Andreas <and@reas.me>
Date:   Thu Apr 22 17:51:08 2021 +0800

    update gitignore and .merlin file

    - also ignore *.bc (our matrix library)
    - fixed .merlin config

commit 7b81aec324d01e87b79d9c3f83ce094160bd1202
Author: Andreas <and@reas.me>
Date:   Thu Apr 22 17:36:03 2021 +0800

    update README

    add Environment and update Usage
    TODO: refine Testing portion

commit 6b0fcdc38f7175e12a30b0bd40d9d8b28998b2cf
Author: Andreas <and@reas.me>
Date:   Thu Apr 22 17:04:36 2021 +0800

    remove locals again....

commit 53b290461d7c5a3a334eced222bf63c118d6297d
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Thu Apr 22 01:08:26 2021 -0400

    added helper function and a few test cases.

commit 67c87df7b686b3555c61f3de98771013316954bb
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Thu Apr 22 00:28:28 2021 -0400

    adding rref for matrix. work in progress.

commit 0dc5c70cb91f37c3d39f6daf5e4c7ef45a1e7c73
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Apr 21 23:07:26 2021 -0400

    added scalar division and multiplication for double scalars

commit 700a39fc496bf4993aec3f868a89f6e7eff71bce
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Apr 21 21:26:56 2021 -0400

    added scalar division and multiplication

    * and / can be used for matrices in addition to regular data types.

commit ebd35388f536a577351f39414efd0592bf0a7090
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Tue Apr 20 23:59:20 2021 -0400

    clean up

commit 0fcb1b0548ec37a313ebfd98179893492ff18ba6
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Tue Apr 20 21:45:49 2021 -0400
```

```
    added dot product.

commit 260a555f38e88b3b07a2afda276c76ceed192040
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Tue Apr 20 12:49:00 2021 -0400

    inverse of a matrix added.

    included three test cases.

commit 6ea07d71de5de4b930a5277a7473e6c76eb4430b
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Tue Apr 20 10:46:50 2021 -0400

    convert int matrix to float matrix.

commit 4198c64df60b396bccde6d31dda648a9ab855146
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Tue Apr 20 00:53:44 2021 -0400

    fixed linking problem for math library

commit 9b0516de7ce7348de758c4b47ffd605ec38a6bd8
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Mon Apr 19 23:14:36 2021 -0400

    adding inverse. work in progress.

commit 55e2a09668f87b63cc62ae44040af5a9d9c619df
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Mon Apr 19 21:47:05 2021 -0400

    matrix determinant is completed.

commit 20836f851dba9cb88a13c76c934cebac938ec4e1
Author: James Ryan <jamesryan@Jamess-MBP-2.fios-router.home>
Date:   Mon Apr 19 18:17:08 2021 -0400

    more fail cases

commit 7aed586b11827bdbe9c60a97007ff25998df44eb
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Mon Apr 19 15:20:28 2021 -0400

    adding determinant. work in progress.

commit 6a0ca10b2672751f20078964798f6246a5b42715
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Mon Apr 19 08:05:40 2021 -0400

    fixed errors from last merge

commit f911b5df5bd4df0617061db6d82dbce0424a4266
Merge: 76a9647 48e2d2e
Author: James Ryan <jamesryan@Jamess-MBP-2.fios-router.home>
Date:   Mon Apr 19 00:56:00 2021 -0400

    Merge branch 'main' of https://github.com/davitbarblishvili/Matcat into main

commit 76a9647ff36ad7ff3829881e8d94d52ab610856d
Author: James Ryan <jamesryan@Jamess-MBP-2.fios-router.home>
Date:   Mon Apr 19 00:55:53 2021 -0400
```

```
    a few more tests

commit 79d2ea50c8de7ab4b06a5d89e299ee05a9f4916b
Author: James Ryan <jamesryan@Jamess-MBP-2.fios-router.home>
Date:   Mon Apr 19 00:53:57 2021 -0400

    minor fixes

commit 48e2d2e726a7ab3e58113f01e10e5cc363b0640b
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Apr 18 20:55:00 2021 -0400

    added test cases and matrix subtraction

commit 049edc8fb024bea58aafc85aa516f86261640d2b
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Apr 18 20:31:19 2021 -0400

    added dot product

commit 881df89245efa9daed9dd7833905b5caddb9ca21
Author: Andreas <and@reas.me>
Date:   Mon Apr 19 02:26:19 2021 +0800

    Add working merlin file

commit ec4cb324829516644bf3fe4739750d2e28852f53
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Apr 18 12:45:37 2021 -0400

    added polymorphism for operators

commit 865b4ce828dca1c7d0d30565a7e5fb3ab0989655
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Thu Apr 15 13:23:55 2021 -0400

    added transpose function

commit 6b71e9fbb3e0907c0efbd0dcafa4bded38fdf62b
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Apr 14 17:16:31 2021 -0400

    Added matrix addition. Compiles but has some linking problems.

    Other functionality is not affected.

commit 033627f40898a69ca743010576af7e205ed09b28
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Apr 14 13:37:42 2021 -0400

    updated matrix printing format

commit 9ee50a34e90d5660df21b70ea0296916d5ba600e
Author: Andreas <and@reas.me>
Date:   Thu Apr 15 01:34:28 2021 +0800

    reduce error msgs for easier debugging

    Notes:
    - some lines to reduce error msgs should be removed
    - fixed a typo in else build_br part in last commit
```

```
commit 52f179a033d17aa0fe1c3159df963f7fffa116c6
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Apr 14 13:21:45 2021 -0400

    fixed matrix printing order

commit 25e65826e97d51d9b1ab4fcdb8d072b1c4922b7b
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Apr 14 13:16:30 2021 -0400

    fixed matrixLibrary.c linking

commit 55c23cc3b4274f177c70363636bd3dcaf8711179
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Tue Apr 13 23:44:38 2021 -0400

    fixed test cases.

commit b25b015477be3c378a138ffbcf09a9cd3f231fbb
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Tue Apr 13 11:20:09 2021 -0400

    create our own Dockerfile

    docker run --rm -it -v `pwd`:/home/matcat -w=/home/matcat columbiasedwards/plt
command line

commit 018798e5b8dc7eb583004f98de5dcfbef87eae00
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Tue Apr 13 09:56:56 2021 -0400

    fixed clang error

commit 1a19e97bdef9bd6f8cfaa58f3102a32f9fee6348
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Tue Apr 13 03:13:41 2021 -0400

    Added Matrix parsing. Added .c file to print matrix.

commit 24945bcfc9e04f7a7ad532b79d9ca793394a2f70
Author: Andreas <and@reas.me>
Date:   Mon Apr 12 20:48:37 2021 +0800

    add first step in fixing assignment problem

    - entirely remove returning multiple values (RMV) feature
    - revert back to banning assignment right after declaration

    TODO: allow assignments like: int n = 100;

commit 3edd427bcdb5bcfcaf443424c3b5429e3754a816
Author: Andreas <and@reas.me>
Date:   Mon Apr 12 20:14:21 2021 +0800

    refine run.sh style

commit 7694538840da907a004cdd22bc34127e485b47b4
Author: Andreas <and@reas.me>
Date:   Mon Apr 12 18:37:12 2021 +0800

    Refine coding environment.

    .merlin:
```

```
    For Ocaml's static analysis (better error messages).
    Partially working (less irrelevant errors in codegen).
    To make this works better, we may need to restructure our repo
    and redefine .merlin.

    run.sh:
    Read desc.

commit 3a38c0a901a754c911b605c15a0947f1caacc294
Author: James Ryan <jamesryan@Jamess-MBP-2.fios-router.home>
Date:   Sun Apr 11 12:29:38 2021 -0400

    adding 0 check to single test script

commit e2d709c049b9d59efcdb225826a93923b36e60e4
Author: James Ryan <jamesryan@Jamess-MBP-2.fios-router.home>
Date:   Sat Apr 10 11:55:08 2021 -0400

    manyPrints fix

commit 81a49b2a83708111e4daf2d70e58487208af88e8
Merge: f6764b7 bc71015
Author: James Ryan <jamesryan@Jamess-MBP-2.fios-router.home>
Date:   Sat Apr 10 11:42:42 2021 -0400

    Merge branch 'main' of https://github.com/davitbarblishvili/Matcat into main

commit f6764b7a54687e78337032cbf688633a34aaddf1
Author: James Ryan <jamesryan@Jamess-MBP-2.fios-router.home>
Date:   Sat Apr 10 11:42:18 2021 -0400

    cleaning some things up

commit bc71015252d3a3e0254e3323edb03f7d9b7daffb
Author: Andreas <and@reas.me>
Date:   Sat Apr 10 06:08:35 2021 +0800

    clean repo; add a common .gitnore

    clean repo: remove .DS_Store
    .gitignore is add to prevent commiting unnecessary files.

commit 715f169913d9a80e5e1481c9504d22faad452eb1
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sat Apr 3 00:14:35 2021 -0400

    fixed some typos

commit 63f2cad0228ae3a9e7b5ddbfb29f71ffbe8f2290
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Fri Apr 2 21:23:43 2021 -0400

    fixed printb

commit e2557aadac9ef4f79a96164c9008ee2bd3a841dd
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Fri Apr 2 20:35:45 2021 -0400

    updated ifelse test case, and added Sif and SReturn

commit 0dc5e27d7d681dc91d19ae13a3aef808bc8f36f7
Author: Andreas <and@reas.me>
Date:   Sat Apr 3 07:38:22 2021 +0800
```

```
    clean _build/.*; update Makefile; update README.md

commit 13681c6b816302b9e9326062b746cce771df849d
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Fri Apr 2 12:50:32 2021 -0400

    added Sassign, SId, SBinop

commit f78209a7284a6147f70a379690c5fb26893b33df
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Fri Apr 2 11:12:54 2021 -0400

    Updated variable declaration.

commit 6dafbc51ce72f0238bd658c3f261eb8baba8a29c
Author: James Ryan <jamesryan@Jamess-MBP-2.fios-router.home>
Date:   Thu Apr 1 20:16:05 2021 -0400

    test edits

commit 259dd5ff0e239acf308a67fcc12f28bc5dffcd22
Author: James Ryan <jamesryan@Jamess-MBP-2.fios-router.home>
Date:   Thu Apr 1 19:54:11 2021 -0400

    some if/else and assignment (bool) tests

commit afe20303abbe5c834213c89eaddf266eb5c47c8d
Author: James Ryan <jamesryan@Jamess-MBP-2.fios-router.home>
Date:   Tue Mar 30 22:58:48 2021 -0400

    filled in some .err; having test-printstring issues

commit 447995e10cf1c42d5b2550bf36ede604be59cdf7
Author: James Ryan <jamesryan@Jamess-MBP-2.fios-router.home>
Date:   Sun Mar 28 14:42:45 2021 -0400

    some for and while tests

commit a7abe80a3b6190180d2f2b98930a48c433e9a283
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 24 22:45:12 2021 -0400

    Adding string to files.

commit 91283273041f5e439bb5d1d0104561ea091f80c9
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 24 21:13:08 2021 -0400

    added some code that was lost due to last merge removal

commit 636e1e0c1467ceaf25ba3b339b9cc6fe54d89c9b
Author: Andreas <and@reas.me>
Date:   Thu Mar 25 08:08:01 2021 +0800

    exclude *.log in Makefile

commit 55cf34097912b9e38b45f8658cf5d560d37e60b8
Author: Andreas <and@reas.me>
Date:   Thu Mar 25 08:05:10 2021 +0800

    Update README
```

```
commit d748b694cdcc36308f6d685bcb9330889af29ee9
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 24 19:55:20 2021 -0400

    Update sast.ml

commit 60ea7c146d30af791ebef5eff638f23d5f979484
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 24 19:50:16 2021 -0400

    Update ast.ml

commit 719daa05e45ac0b73efc74d30e962cd604a2baa1
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 24 19:45:03 2021 -0400

    removed last merge

commit cabe77e5d7d2976e65e628a5beaa9f45c985b027
Merge: d51c44f 4f1bc9f
Author: James Ryan <jamesryan@Jamess-MBP-2.fios-router.home>
Date:   Wed Mar 24 19:21:57 2021 -0400

    Merge branch 'main' of https://github.com/davitbarblishvili/Matcat into main

commit d51c44f912a17233546cbc5e77f32a4a2228cd40
Author: James Ryan <jamesryan@Jamess-MBP-2.fios-router.home>
Date:   Wed Mar 24 19:21:39 2021 -0400

    pretty print fix

commit 4f1bc9ff98ee80312e9a32ac14155c022c57be4d
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 24 16:10:46 2021 -0400

    cleaned up some filed; updated name declarations.

commit 143197b17d5a96e6cde576a3a6263d20bfa54396
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 24 15:09:32 2021 -0400

    changed Literal to IntLit to be consisent in all files

commit fab623d5e33004c2b7d526611c023eb93792fe9a
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 24 11:56:11 2021 -0400

    updated Makefile

commit 2f61ef193e3053c102410ec2eca55b1d39ef0260
Author: Andreas <and@reas.me>
Date:   Wed Mar 24 02:25:39 2021 +0800

    add tarbar builder

commit e0fbad8ae5526b837e491e6fb44157602e58a2be
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Tue Mar 23 01:09:33 2021 -0400

    updated semant and codegen

    defined each print function for each data type.
```

```
commit 66a0e9fb574e5880b630c23fdb9ddbe8e639375d
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Mon Mar 22 22:18:37 2021 -0400

    updated Matcat

    1) fixed test cases
    2) reduced codegen to the point where it only prints hello world.
    3) added test to Makefile.
    4) Added testall.sh

commit 2480a4d2d56a0ff85be378415403c5df9ca49f59
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Mon Mar 22 20:00:11 2021 -0400

    removed some functions.

commit e8a29430a96df727e132a87dd5ddef9ed610e178
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Mon Mar 22 19:47:06 2021 -0400

    Create codegen_temp.ml

    working temp file for code_gen to print hello world.

commit c117c39bb7e33f7758074911f8b2c69e0ce29c84
Author: Andreas <and@reas.me>
Date:   Tue Mar 23 01:06:41 2021 +0800

    add simple test.sh that run one .mc file

commit f1240901afdc4ed94a28ea876482386c353aca9a
Author: Andreas <and@reas.me>
Date:   Tue Mar 23 00:51:06 2021 +0800

    Add/refine/sign a few more testcases

commit ca6ec862216dc4db903798f9d0829342b8c1a48d
Author: Andreas <and@reas.me>
Date:   Tue Mar 23 00:50:34 2021 +0800

    add semant, sast, codegen for helloworld

    For now, the print function can only print Integer (same as microc)

    Check TODOs in those files for parts to be fixed

commit 77c78fa545da040a086b998bfcc730813bd90106
Author: Andreas <and@reas.me>
Date:   Tue Mar 23 00:45:27 2021 +0800

    refine pass-helloworld

commit 9678e30ff594d6f166fb4617ad4707be4d73d95d
Author: Andreas <and@reas.me>
Date:   Mon Mar 22 20:57:03 2021 +0800

    Refine styling; Add missing string_of_xxxs in ast

    Refine var names; Align/pad brackets

    code consistancy fix:
    types -> data_types
```

```
    grammar fix:
    string_of_operators -> string_of_operator
    (The string only contains one operator)

commit f617baeb3da1b29f89d56b3a1192997a3ccd3a19
Author: Andreas <and@reas.me>
Date:   Mon Mar 22 17:54:09 2021 +0800

    Clean repo: I shouldn't commit this file 🤦

commit 01aa7a528f9f6b6df35de86746fa1a59986ee5cd
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sat Mar 20 22:05:02 2021 -0400

    Update Makefile

commit 1f5973206c9bf1d03030b842f81973e26ceb228b
Author: mk4069 <55564448+mk4069@users.noreply.github.com>
Date:   Sat Mar 20 21:43:49 2021 -0300

    Update ast.ml

commit 23c89f0af11b75ab50913871d6706752806969ad
Merge: 9a0e33a c434f91
Author: mk4069 <55564448+mk4069@users.noreply.github.com>
Date:   Sat Mar 20 21:42:30 2021 -0300

    Merge branch 'main' of https://github.com/davitbarblishvili/Matcat into main

commit 9a0e33a88ea57111c6c9c2d3d88bea633778982b
Author: mk4069 <55564448+mk4069@users.noreply.github.com>
Date:   Sat Mar 20 21:42:02 2021 -0300

    Update parser.mly

commit c434f91dd0d9d0cc2984602587ee0eaca751f24f
Author: Andreas <and@reas.me>
Date:   Sun Mar 21 04:36:47 2021 +0800

    Uses parser.mly for parsing

commit 349c46760d94c89c66fe4c3b204143035c3816c7
Author: Andreas <and@reas.me>
Date:   Sun Mar 21 02:43:13 2021 +0800

    uses parser.mly as the default name for the parser

    + ocamlyacc parser.mly
    6 rules never reduced

    This can be fixed after we uncomment them
    but b4 that, we need to update the ast for them

commit a7033980e0e2addb77cdc7557558b8da07edd5ee
Author: Andreas <and@reas.me>
Date:   Sat Mar 20 21:32:08 2021 +0800

    add one simple fail testcase

commit 268545b42b452a836c4f4345ea5d4e1c17ea6e5a
Author: Andreas <and@reas.me>
Date:   Sat Mar 20 21:29:13 2021 +0800
```

```
    add one simple test case.

commit f13cd457e3b15337124172e46f5a1c25f5a3053b
Author: Andreas <and@reas.me>
Date:   Sat Mar 20 21:24:27 2021 +0800

    add make matcat.native (for easier debugging)

    some parts should be uncommented to continue the debugging

    We can debug our parser/scanner works b4 writing the matrix/vector parts

commit 32eb19ccda8edfd2dec7ba9b6d17ac7af5f06209
Author: mk4069 <55564448+mk4069@users.noreply.github.com>
Date:   Fri Mar 19 15:02:43 2021 -0300

    Update ast.ml

commit a9c7381289bdd7a77b5b5b004d47d6e5635788aa
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 17 22:48:20 2021 -0400

    Update ast.ml

commit 0299a9ec7db033b4ae0e2a22cef853be1f204a04
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 17 22:47:31 2021 -0400

    Update scanner.mll

commit 34daffbd29389389522fc6bf8fa269e3c8500561
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 17 22:47:07 2021 -0400

    Rename Scanner.mll to scanner.mll

commit ff2ce74ea47a663baf60416b65e596f450eb247f
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 17 22:46:41 2021 -0400

    Rename Matcat_Overview.txt to overview.txt

commit 244305dbb14c2348008f7e8f02f5abb6736ed149
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 17 22:46:28 2021 -0400

    Rename Matcat_Parser.mly to parser.mly

commit 76c69685330682e361f1f1fe8ca7b38654622e74
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 17 22:46:14 2021 -0400

    Rename Matcat_ast.ml to ast.ml

commit 3c1f50a8aeef21dd49fc264f6c073591d849c085
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 17 22:44:57 2021 -0400

    Create Matcat_ast.ml

commit e92282e4f59acf70b828b9625894cdc4d30c35fe
Author: Davit Barblishvili <db3230@columbia.edu>
```

```
Date:   Wed Mar 17 22:24:26 2021 -0400

    changed float to double

commit 15807a88318177324b6292ad2942c5d225714e23
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 17 22:10:59 2021 -0400

    Delete test

commit 65b1cf8a3e40f55479010ee2de1b7cd29d2ae640
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 17 22:10:38 2021 -0400

    Add files via upload

commit 04e3b752e832db8fde3dbeef1a9832ab39728400
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 17 22:09:21 2021 -0400

    Create test

commit 5008588d0875f16d2b03776f9ff57cbce6ed2d39
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 17 22:07:24 2021 -0400

    Delete tests

commit ffcaa5fb1d50fa342b9d4cde7d8637aeeadd0f76
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 17 22:07:06 2021 -0400

    Create tests

commit 3b6f80deeeaad48d52b2c567d2e99194273aa29f
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Mar 17 21:55:55 2021 -0400

    Delete tests directory

commit 86faebe274b213ab29a35dc1954d425e69d73c06
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Mon Mar 8 19:55:33 2021 -0500

    fixed typo

commit 96d2a2b29362ba33dbd86c8a647205c7a101f39f
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Mon Mar 8 19:52:22 2021 -0500

    scanner.mll

commit 0f1ab285ee68888961b2d813cd09103c29cf9e1a
Author: Andreas <and@reas.me>
Date:   Thu Feb 25 08:47:11 2021 +0800

    clean up parser; revert back to matrix[x][y]

commit e72747952370abef1f18d2018e628131b2b802c9
Author: Andreas <and@reas.me>
Date:   Thu Feb 25 03:18:10 2021 +0800

    add rules for vector/matrix element extractors
```

```
    vector v = [1,2;];
    int elm = v[1];

    matrix m = <1,2,3;4,5,6>;
    int elm = m[1,2];
    int elm2 = <1,2,3;4,5,6>[1,2];

commit 44865fa0976a5e6ad793e6d229f2b418a917d7e7
Author: Andreas <and@reas.me>
Date:   Thu Feb 25 03:17:05 2021 +0800

    add make clean

commit 6b44b0225c24e8f798e736d6805e24d55af7c066
Merge: 5bf1179 d2d0f0f
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Tue Feb 23 22:04:26 2021 -0500

    Merge branch 'main' of https://github.com/davitbarblishvili/Matcat into main

commit 5bf11794811489974337135eb2067daa32088547
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Tue Feb 23 22:03:39 2021 -0500

    changed vector and matrix formatting to fix errors

commit d2d0f0f6697e272680a342dc4ed1646b22ca1f2c
Author: Andreas <and@reas.me>
Date:   Wed Feb 24 01:39:16 2021 +0800

    clean repo: remove parser.ml and parser.mli

commit 5672d66f39f3dfa008cddae2156c3f136c321b00
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Mon Feb 22 20:39:02 2021 -0500

    added transpose and inverse as unop

commit f682a112b25fdd4b61d6a5570f174726a9fce71d
Author: James Ryan <jamesryan@Jamess-MBP-2.fios-router.home>
Date:   Mon Feb 22 20:02:46 2021 -0500

    compiles somehow

commit d15263135577cd07dee13215ed91744d3cdd4f08
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:11:39 2021 -0500

    Rename codegen.ml to MicroC/codegen.ml

commit 64adf8f412201f96e41cbd883c0b1e2f5a6d5ef2
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:10:28 2021 -0500

    Rename font2c to MicroC/font2c

commit e9fd2da5bc50fb7215bed416195e7b0d972754ab
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:10:12 2021 -0500

    Rename microc.ml to MicroC/microc.ml
```

```
commit 11be4390d5a9fde780c18e80ed37c53086a607f5
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:09:59 2021 -0500

    Rename microcparse.mly to MicroC/microcparse.mly

commit da5b1d03e19f1f55eca89440e07df79be15df3c3
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:09:47 2021 -0500

    Rename printbig.c to MicroC/printbig.c

commit 9634ab5810d406f6e64b747bb0ad913bc04ebf00
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:09:26 2021 -0500

    Delete printbig.o

commit a4d6951cba7a9c1f3d3c6e4a7e0407f37d6795e2
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:09:01 2021 -0500

    Rename ast.ml to MicroC/ast.ml

commit 3a9ab5f1e606981ca12338a2b52d6771dec8495c
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:08:14 2021 -0500

    Delete test-gcd.out

commit 97ee7ac62496ce2b4f06e400bc3bb4495844f8e8
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:08:04 2021 -0500

    Rename sast.ml to MicroC/sast.ml

commit 9faade9fcb0eee6f3f2c76470ec412fd587f2e43
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:07:48 2021 -0500

    Rename scanner.mll to MicroC/scanner.mll

commit eec16990c17c0c819122e69937186fd89c8f6e11
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:07:33 2021 -0500

    Rename semant.ml to MicroC/semant.ml

commit 5cf43cf66d8acfbf22b6114c16ead6157c77745c
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:07:21 2021 -0500

    Delete test-gcd.exe

commit f0910b5011d66da8697247a2e2213fb962c8a2a5
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:07:01 2021 -0500

    Rename test-gcd.out to MicroC/test-gcd.out

commit 5f49f8829799cb60fb50a0272caa177f72a9ef0f
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:06:41 2021 -0500
```

```
    Rename MicrcoC/testall.sh to MicroC/testall.sh

commit 58960e0da37b09af6f4d6c26ed3ae511dea453e0
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:06:15 2021 -0500

    Rename testall.sh to MicrcoC/testall.sh

commit f67a1e1c6575fe61e546945091e1bd266fed9786
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:05:57 2021 -0500

    Rename arcade-font.pbm to MicroC/arcade-font.pbm

commit 4f5bbc4fbf53584a3dd8694c56ac080689deb774
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:04:35 2021 -0500

    Rename _tags to MicroC/_tags

commit e754a59bc9ca2a9d58df94fbca9d2181a5a2923e
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:04:15 2021 -0500

    Rename README to MicroC/README

commit e5025393bd19d0de30ff866ec3ad530b85fd1170
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:03:56 2021 -0500

    Rename Makefile to MicroC/Makefile

commit beaf5b5044ee9bc9f441c6fd48c3d27594c3303a
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 23:03:13 2021 -0500

    Rename Dockerfile to MicroC/Dockerfile

commit 476eaa13a5939f4a93eb1632a3868222d2e1015b
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 22:22:43 2021 -0500

    fixed "undefined entry" problem

commit 7469dda904035d041d2b598a226937abb765957c
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 22:10:24 2021 -0500

    fixed some typos

commit 0dfaba06ca5061a08840856ab824dc4b37de880c
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 22:04:35 2021 -0500

    Update Matcat_Parser.mly

commit 824c42ffaf0bd8e976b1e336ad1a147764ab834b
Author: mk4069 <55564448+mk4069@users.noreply.github.com>
Date:   Sun Feb 21 23:58:27 2021 -0300

    Update Matcat_Parser.mly
```

```
commit 41f82cd2928db46379b9a95b784e4ac6b0063c57
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 21:01:37 2021 -0500

    Update Matcat_Parser.mly

commit 37599c06b15ee613be428f2f897cc72bfa9d7b0b
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 20:43:45 2021 -0500

    Update Matcat_Parser.mly

commit 2541f753894b374d4bd1acfa6cf0ab3926d844f6
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Sun Feb 21 20:16:58 2021 -0500

    Update Matcat_Parser.mly

commit f661c36f7807308f61238179f14da9bbe0ee9f66
Author: mariamkhmaladze <73886449+mariamkhmaladze@users.noreply.github.com>
Date:   Sun Feb 21 22:00:28 2021 -0300

    Add files via upload

    Parser

commit 0e29cf4d8e35a89967c217135434c1feb92a7158
Author: Andreas <and@reas.me>
Date:   Sat Feb 20 18:20:34 2021 +0800

    add microc as base starter;

commit 11dbd6e33e0e73247ccab8218b602461bf0d5ca0
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Feb 17 21:27:23 2021 -0500

    Update Matcat_Overview.txt

commit 8e454dfee7ea2e01ea961a3c73d826c176097f34
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Feb 17 21:24:27 2021 -0500

    Update Matcat_Overview.txt

commit 693d5a7b9c1c125c4781867d5d9377fd4c8daeb5
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Feb 17 21:20:53 2021 -0500

    Update Matcat_Overview.txt

commit c0cfe016fb67e6c40c8de234c84f611ee181ea93
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Feb 17 21:16:29 2021 -0500

    Add files via upload

    Overview for Matcat

commit f605b6d08f35cdea667333ae2baf439a5a1b6605
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Feb 17 21:06:19 2021 -0500

    Update README.md
```

```
commit 82e9168ee3ca8de2b2ec854cea771b0f77dbd775
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Feb 17 21:05:06 2021 -0500

    overview of Matcat

commit 2cc448b91e8524276d47edef52b217ae9b3ecc2e
Author: Davit Barblishvili <db3230@columbia.edu>
Date:   Wed Feb 17 21:04:27 2021 -0500

    Initial commit
```

10.3 matcat.ml

```
(*
Date        Author        Changes
2021-03-22  Andreas       Add basic parts based on MicroC to make hello world works
*)

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: ./matcat.native [-a|-s|-l|-c] [file.mc]" 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)
```

10.4 scanner.mll

```
(* Authors: Matcat Team *)
(* Ocamllex scanner for Matcat *)

{ open Parser }

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

```
rule token = parse
  [' ' '\t' '\r' '\n']  { token lexbuf } (* Whitespace *)
| "/*"                  { multi_comment lexbuf }           (* Comments *)
| "//"                  { short_comment lexbuf }
| '('                   { LPAREN }
| '{'      { LBRACE }
| ')'      { RPAREN }
| '}'      { RBRACE }
| '['      { LBRACK}
| ']'      { RBRACK}
| '^'      { CIRCU }
| ';'      { SEMI }
| ','      { COMMA }
| '+'      { PLUS }
| '-'      { MINUS }
| '*'      { TIMES }
| '/'      { DIVIDE }
| '='      { ASSIGN }
| ':'      { COLON }
| "cr"     { CR }
| "dot"    { DOT }
| "=="     { EQ }
| "!="     { NEQ }
| '<'      { LT }
| "<="     { LEQ }
| ">"      { GT }
| ">="     { GEQ }
| "&&"     { AND }
| "||"     { OR }
| "!"      { NOT }
| "if"     { IF }
| "else"   { ELSE }
| "for"    { FOR }
| "while"  { WHILE }
| "return" { RETURN }
| "func"   { FUNC }
| "int"    { INT }
| "bool"   { BOOL }
| "matrix" {MATRIX}
| "double"  { DOUBLE }
| "void"   { VOID }
| "true"   { BLIT(true)  }
| "false"  { BLIT(false) }
| '"' ([^ '"']* as lit) '"' { STRINGLIT(lit) }
| digits as lxm { INTLIT(int_of_string lxm) }
| digits '.'  digit* ( ['e' 'E'] ['+' '-']? digits )? as lxm { DOUBLELIT(lxm) }
| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']*     as lxm { ID(lxm) }
| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']*  as lxm { ID(lxm) }
| eof { EOF }
| _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }

and multi_comment = parse
             "*/" { token lexbuf }
             (* Return to normal scanning *)
             | _ { multi_comment lexbuf }

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

10.5 parser.mly

```
/* Author: Matcat Team */
/* Ocamlyacc parser for Matcat */

%{
open Ast
%}


%token SEMI LPAREN RPAREN LBRACE RBRACE COMMA PLUS MINUS TIMES DIVIDE ASSIGN RBRACK
LBRACK COLON CIRCU
%token STRUCT
%token NOT EQ NEQ LT LEQ GT GEQ TRUE FALSE AND OR
%token RETURN IF ELSE FOR WHILE INT STRING BOOL DOUBLE VOID
/* added */
%token CR DOT MATRIX FUNC
%token <int> INTLIT
%token <bool> BLIT
%token <string> ID DOUBLELIT
%token <string> STRINGLIT
%token EOF

%start program
%type <Ast.program> program

%nonassoc NOELSE /* above? */
%nonassoc ELSE
%right ASSIGN
%left OR
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE
%left CR DOT
%right NOT
%right CIRCU
%left

%%


program:
  decls EOF { $1 }

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

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

fdecl:
   FUNC ID LPAREN formals_opt RPAREN typ LBRACE stmt_list RBRACE
     { {
        fname = $2;
        formals = List.rev $4;
        data_type =$6;
        locals = [];
        body = List.rev $8 } }
```

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

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

/* added matrix in types */
typ:
    INT     { Int    }
  | BOOL    { Bool   }
  | DOUBLE  { Double }
  | VOID    { Void   }
  | MATRIX  { Matrix }
  | STRING  { String }

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

stmt:
    expr SEMI                                  { Expr $1                    }
  | RETURN expr_opt SEMI                       { Return $2                  }
  | LBRACE stmt_list RBRACE                    { Block(List.rev $2)         }
  | IF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5, Block([]))       }
  | IF LPAREN expr RPAREN stmt ELSE stmt    { If($3, $5, $7)             }
  | FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN stmt
                                             { For($3, $5, $7, $9)        }
  | WHILE LPAREN expr RPAREN stmt           { While($3, $5)              }
  | typ ID SEMI                              { Vdecl(($1, $2), Noassign)  }
  | typ ID ASSIGN expr SEMI                  { Vdecl(($1, $2), $4)        }

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

expr:
    INTLIT           { IntLit($1)            }
  | DOUBLELIT       { DoubleLit($1)          }
  | BLIT             { BoolLit($1)           }
  | STRINGLIT        { StringLit($1)         }
  | TRUE             { BoolLit(true)         }
  | FALSE            { BoolLit(false)        }
  | ID               { Id($1)                }
  | expr PLUS   expr { Binop($1, Add,   $3)  }
  | expr MINUS  expr { Binop($1, Sub,   $3)  }
  | expr TIMES  expr { Binop($1, Mult,  $3)  }
  | expr DIVIDE expr { Binop($1, Div,   $3)  }
  | expr EQ     expr { Binop($1, Equal, $3)  }
  | expr NEQ    expr { Binop($1, Neq,   $3)  }
  | expr LT     expr { Binop($1, Less,  $3)  }
  | expr LEQ    expr { Binop($1, Leq,   $3)  }
  | expr GT     expr { Binop($1, Greater, $3) }
  | expr GEQ    expr { Binop($1, Geq,   $3)  }
  | expr AND    expr { Binop($1, And,   $3)  }
  | expr OR     expr { Binop($1, Or,    $3)  }
  | MINUS expr %prec NOT { Unop(Neg, $2)     }
  | NOT expr         { Unop(Not, $2)         }
  | ID ASSIGN expr   { Assign($1, $3)        }
  | ID LPAREN args_opt RPAREN { Call($1, $3) }
  | LPAREN expr RPAREN { $2                  }
```

```
  | expr CR expr     { Binop($1,Cr,$3)        }
  | expr DOT expr    { Binop($1,Dot,$3)       }
  | LBRACK matrix_value RBRACK        { MatrixLit($2)                    }
  | ID LBRACK expr RBRACK LBRACK expr RBRACK { MatrixAccess($1, $3, $6) }
  | ID LBRACK expr COMMA COLON RBRACK { MatrixAccess1D($1, $3)           }
  | ID LBRACK COLON COMMA expr RBRACK { MatrixAccessCol($1, $5)          }
  | ID CIRCU expr                     { MatrixPower($1, $3)              }
  | ID LBRACK COLON COMMA COLON RBRACK { MatrixDiagonal($1)          }

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

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

matrix_value:
  LBRACK args_opt RBRACK                    { [MatrixLit(List.rev $2)]   }
| LBRACK args_opt RBRACK COMMA matrix_value { MatrixLit(List.rev $2)::$5 }
| LBRACK args_opt RBRACK matrix_value       { MatrixLit(List.rev $2)::$4 }
```

10.6 ast.ml

```
(*
Author: Matcat team
Copyright 2021, Matcat
*)

type operator = Add | Sub | Mult | Div | Equal
              | Neq | Less | Leq | Greater
              | Geq | And | Or | Cr | Dot


type unary_operator = Not | Neg

type data_type = Int | Double | String | Void | Bool | Matrix

type bind = data_type * string

type expr =
  IntLit of int
| Binop of expr * operator * expr
| Unop of unary_operator * expr
| DoubleLit of string
| StringLit of string
| BoolLit of bool
| MatrixLit of expr list
| Id of string
| Assign of string * expr
| Call of string * expr list
| MatrixAccess of string * expr * expr
| MatrixAccess1D of string * expr
| MatrixAccessCol of string * expr
| MatrixDiagonal of string
| MatrixPower of string * expr
| Noexpr
| Noassign

type global = bind * expr
```

```
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
| Vdecl of bind * expr


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

type program = global list * func_decl list


let string_of_operator = function
  Add -> "+"
| Sub -> "-"
| Mult -> "*"
| Div -> "/"
| Equal -> "=="
| Neq -> "!="
| Less -> "<"
| Leq -> "<="
| Greater -> ">"
| Geq -> ">="
| And -> "&&"
| Or -> "||"
| Cr -> "cr"
| Dot -> "dot"

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


let string_of_data_type = function
    Int -> "int"
  | Bool -> "bool"
  | Double -> "double"
  | String -> "string"
  | Matrix -> "matrix"
  | Void -> "void"

(* Pretty-printing functions *)
let string_of_bind bind =
  string_of_data_type (fst bind) ^ " " ^ (snd bind)

let rec string_of_expr = function
      IntLit (l) -> string_of_int l
    | Binop(e1, o, e2) ->
      string_of_expr e1 ^ " " ^ string_of_operator o ^ " " ^ string_of_expr e2
    | Unop(o, e) -> string_of_uop o ^ string_of_expr e
    | DoubleLit(l) -> l
    | StringLit(l) -> l
    | BoolLit(true) -> "true"
    | BoolLit(false) -> "false"
```

```
    | MatrixLit(l) ->  "matrixLit[" ^ String.concat ", " (List.map string_of_expr l) ^
"]"
    | Id(s) -> s
    | Assign(v, e) -> v ^ " = " ^ string_of_expr e
    | MatrixAccess(s,e1,e2)-> "MatrixAccess " ^ s ^ "[" ^ string_of_expr(e1) ^ "]" ^
"[" ^ string_of_expr(e2) ^ "]"
    | MatrixAccess1D(s,e1)-> "MatrixAccess1D " ^ s ^ "[" ^ string_of_expr(e1) ^ ",:]"
    | MatrixAccessCol(s,e1)-> "MatrixAccessCol " ^ s ^ "[:," ^ string_of_expr(e1) ^
"]"
    | MatrixDiagonal(s)-> "MatrixDiagonal " ^ s ^ "[:, :]"
    | MatrixPower(s,e1) -> "MatrixPower " ^ s ^"^" ^ string_of_expr(e1)
    | Call(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
    | Noexpr -> "(Noexpr)"
    | Noassign -> "(Noassign)"

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
  | Vdecl(b, e) -> match e with
      Noassign -> string_of_bind b ^ ";\n"
    | _       -> string_of_bind b ^" = "^ string_of_expr e ^ ";\n"

  (*variable declaration*)
let string_of_vdecl (t, id) = string_of_data_type t ^ " " ^ id ^ ";\n"

let string_of_formals formals = List.map string_of_bind formals

let string_of_fdecl fdecl =
  "func " ^
  fdecl.fname ^ "(" ^ String.concat ", " (string_of_formals fdecl.formals) ^
  ")" ^ string_of_data_type fdecl.data_type ^ "\n{\n" ^
  String.concat "" (List.map string_of_stmt fdecl.body) ^
  "}\n"


let string_of_global (b, e) =
  if string_of_expr e = ""
  then string_of_bind b ^ ";\n"
  else string_of_bind b ^" = "^ string_of_expr e ^ ";\n"


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

10.7 sast.ml

```
(* Authors: Matcat Team *)

open Ast
```

```
type sexpr = data_type * sx
and sx =
    SIntLit of int
  | SBinop of sexpr * operator * sexpr
  | SUnop of unary_operator * sexpr
  | SDoubleLit of string
  | SStringLit of string
  | SBoolLit of bool
  | SMatrixLit of sexpr list * int * int
  | SId of string
  | SAssign of string * sexpr
  | SCall of string * sexpr list
  | SMatrixAccess of string * sexpr * sexpr
  | SMatrixAccess1D of string * sexpr
  | SMatrixAccessCol of string * sexpr
  | SMatrixDiagonal of string
  | SMatrixPower of string * sexpr
  | SNoexpr
  | SNoassign

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
  | SVdecl of bind * sexpr

type sfunc_decl = {
    sfname : string;
    sformals : bind list;
         sdata_type : data_type;
    slocals : bind list;
    sbody : sstmt list;
  }

type sprogram = global list * sfunc_decl list


(* Pretty-printing functions *)

let rec string_of_sexpr (t, e) =
  "(" ^ string_of_data_type t ^ " : " ^ (match e with
      SIntLit(l) -> string_of_int l
    | SBinop(e1, o, e2) ->
        string_of_sexpr e1 ^ " " ^ string_of_operator o ^ " " ^ string_of_sexpr e2
    | SUnop(o, e) -> string_of_uop o ^ string_of_sexpr e
    | SDoubleLit(l) -> l
    | SStringLit(l) -> l
    | SBoolLit(true) -> "true"
    | SBoolLit(false) -> "false"
    | SMatrixLit(l, r, c) -> "rows: " ^ string_of_int r ^ ", cols: " ^
                  string_of_int c ^ " : [" ^ String.concat ", " (List.map
string_of_sexpr l) ^ "]"
    | SId(s) -> s
    | SAssign(v, e) -> v ^ " = " ^ string_of_sexpr e
    | SMatrixAccess(s,e1,e2)-> "SMatrixAccess " ^ s ^ "[" ^ string_of_sexpr(e1) ^ "]"
^ "[" ^ string_of_sexpr(e2) ^ "]"
    | SMatrixAccess1D(s,e1)-> "SMatrixAccess1D " ^ s ^ "[" ^ string_of_sexpr(e1) ^
",:]"
    | SMatrixAccessCol(s,e1)-> "SMatrixAccessCol " ^ s ^ "[:," ^ string_of_sexpr(e1) ^
"]"
```

```
    | SMatrixDiagonal(s)-> "MatrixDiagonal " ^ s ^ "[:, :]"
    | SMatrixPower(s,e1) -> "SMatrixPower " ^ s ^"^" ^ string_of_sexpr(e1)
    | SCall(f, el) ->
        f ^ "(" ^ String.concat ", " (List.map string_of_sexpr el) ^ ")"
    | SNoexpr -> ""
    | SNoassign -> ""
  )
  ^ ")"

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
  | SVdecl(b, e) ->
    if string_of_sexpr e = "" then
        string_of_data_type (fst b) ^ " " ^ (snd b) ^ ";\n"
    else string_of_data_type (fst b) ^ " " ^ (snd b) ^" = "^ string_of_sexpr e ^ ";\n"



    let string_of_sfdecl fdecl =
      "func " ^
      fdecl.sfname ^ "(" ^ String.concat ", " (List.map snd fdecl.sformals) ^
      ")" ^ string_of_data_type fdecl.sdata_type ^ "\n{\n" ^
      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)
```

10.8 semant.ml

```
(* Authors: Matcat Team *)

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) =

    let check_binds (kind : string) (to_check : bind list) =
      let check_it checked binding =
        let void_err = "illegal void " ^ kind ^ " " ^ snd binding
        and dup_err = "duplicate " ^ kind ^ " " ^ snd binding
        in match binding with
          (* No void bindings *)
```

```
        (Void, _) -> raise (Failure void_err)
      | (_, n1) -> match checked with
                     (* No duplicate bindings *)
                       ((_, n2) :: _) when n1 = n2 -> raise (Failure dup_err)
                     | _ -> binding :: checked
    in let _ = List.fold_left check_it [] (List.sort compare to_check)
        in to_check
  in

  let check_globals (kind : string) (to_check : global list) =

    let check_it (checked, global_map) (binding, ex) =
      let void_err = "illegal void " ^ kind ^ " " ^ snd binding
      and dup_err = "duplicate " ^ kind ^ " " ^ snd binding

    in



    (* Return a semantically-checked expression, i.e., with a type *)
    let rec expr = function
    IntLit l -> (Int, SIntLit l)
  | DoubleLit l -> (Double, SDoubleLit l)
  | BoolLit l  -> (Bool, SBoolLit l)
  | Noexpr     -> (Void, SNoexpr)
  | StringLit l -> (String, SStringLit l)
  (* TODO: CHECK MATRIX LITERAL IN GLOBAL*)
  | Unop(op, e) as ex ->
      let (t, e') = expr e in
      let ty = match op with
        Neg when t = Int || t = Double -> t
      | Not when t = Bool -> Bool
      | _ -> raise (Failure ("illegal unary operator " ^
                             string_of_uop op ^ string_of_data_type 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
      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 = Double -> Double
        | Equal | Neq            when same               -> Bool
        | Less | Leq | Greater | Geq
                 when same && (t1 = Int || t1 = Double) -> Bool
        | And | Or when same && t1 = Bool -> Bool
        | Add | Sub  | Mult when same && t1 = Matrix -> Matrix
        | Mult when t1 = Int && t2 = Matrix -> Matrix
        | Mult when t1 = Double && t2 = Matrix -> Matrix
        | Dot when same && t1 = Matrix -> Double
        | Div when t1 = Matrix && t2 = Int -> Matrix
        | Div when t1 = Matrix && t2 = Double -> Matrix
        | _ -> raise (
                  Failure ("illegal binary operator " ^
                                string_of_data_type t1 ^ " " ^ string_of_operator op
^ " " ^
                                string_of_data_type t2 ^ " in " ^ string_of_expr e))
                    in (ty, SBinop((t1, e1'), op, (t2, e2')))

  | _ -> raise (Failure ("illegal operation for global initialization"))
      in
```

```
        let (tx, ex') = expr ex

        in match (binding, ex) with
          (* No void bindings *)
          ((Void, _), _) -> raise (Failure void_err)
        | ((t1, n1), _) -> if StringMap.mem n1 global_map then raise (Failure dup_err)
else
                           let global_map = StringMap.add (snd binding) ((fst binding),
tx) global_map in
                           if tx = t1 || tx = Void then (((binding, (tx, ex')) ::
checked), global_map)
                         else raise (Failure ("unmatch types " ^ string_of_data_type t1
^ " and " ^ string_of_data_type tx))

      in let _ = List.fold_left check_it ([], StringMap.empty) to_check
         in to_check
    in

    (**** Checking Global Variables ****)
    let globals' = check_globals "global" globals in
    let glob =
        let extract (fst, _) = fst
        in List.map extract globals'
    in

    (**** 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 {
        data_type = ret;
        fname = name;
        formals =
      (let rec create_ty_list = (function
        [] -> []
        | hd::tl -> (hd, "x")::(create_ty_list tl))
        in create_ty_list ty);
        locals = []; body = [] } map
      in List.fold_left add_bind StringMap.empty [ ("print", [Int],Void);
                                   ("printb", [Bool],Void);
                                   ("printd", [Double],Void);
                                   ("printm", [Matrix],Void);
                                   ("printStr", [String],Void);
                                   ("matrxAdd", [Matrix; Matrix], Matrix);
                                   ("transpose", [Matrix], Matrix);
                                   ("det", [Matrix], Double);
                                   ("inv", [Matrix], Matrix);
                                   ("rref", [Matrix], Matrix);
                                   ("isInv", [Matrix], Bool);
                                   ("accessMatrix",[Matrix;Int;Int],Double);
                                   ("accessMatrix1D",[Matrix;Int],Matrix);
                                   ("accessMatrixCol",[Matrix;Int],Matrix);
                                   ("get_diagonal",[Matrix],Matrix);
                                   ("rotate90",[Matrix],Void);
                                   ("check_symmetry",[Matrix],Int)]
    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 are void or duplicates *)
      let formals' = check_binds "formal" func.formals in
      (* 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

    let rec find_locals stmt_list locals= match stmt_list with
      | Vdecl (b, _):: tl -> b :: (find_locals tl locals)
      | Block sl :: tl ->
        let rec find_locals_in_block sl locals= match sl with
              | Block sl :: tl  -> find_locals_in_block (sl @ tl) locals(* Flatten
blocks *)
              | Vdecl(b, _) :: tl  -> (b :: (find_locals tl locals))
              | _ ::tl       -> find_locals_in_block tl locals
              | [] -> locals
              in (find_locals_in_block sl locals) @ (find_locals tl locals)
      | If(_,_,e) :: tl -> (find_locals [e] locals) @ (find_locals tl locals)
      | For(_,_,_,e) ::tl -> (find_locals [e] locals) @ (find_locals tl locals)
      | While(_,e) :: tl -> (find_locals [e] locals) @ (find_locals tl locals)
      | _ :: tl-> find_locals tl locals
      | [] -> locals
      in

    let locals = find_locals func.body [] in
    let locals' = check_binds "local" locals in
    let symbol = List.fold_left (fun m (ty, name) -> StringMap.add name ty m)
                    StringMap.empty (glob @ formals') in
      let symbols = [symbol] in
    let type_of_identifier s symbols =
      (* try StringMap.find s (List.hd symbols) with Not_found -> raise (Failure
("undeclared identifier " ^ s)) *)
      let rec f list = match list with
        [] -> raise (Failure ("undeclared identifier " ^ s))
      | fst :: tail -> try StringMap.find s fst with Not_found -> f tail
    in f symbols
    in


      let rec get_dims = function
      MatrixLit l -> List.length l :: get_dims (List.hd l)
     | _ -> []
```

```
      in

      let rec flatten d = function
      [] -> []
      | MatrixLit hd::tl -> if List.length hd != List.hd d then raise
(Failure("Invalid dims")) else List.append (List.rev (flatten ((List.tl) d) hd))
(flatten d tl)
      | a -> a
      in

      (* Return a semantically-checked expression, i.e., with a type *)
      let rec expr e symbols = match e with
          IntLit l -> (Int, SIntLit l)
        | DoubleLit l -> (Double, SDoubleLit l)
        | BoolLit l  -> (Bool, SBoolLit l)
        | Noexpr     -> (Void, SNoexpr)
        | Noassign     -> (Void, SNoassign)
        | StringLit l ->(String, SStringLit l)
        | MatrixLit l ->
          let d = get_dims (MatrixLit l) in
          let rec all_match = function
            [] -> ignore()
            | hd::tl -> if tl != [] then
                          let (t1, _) = expr hd symbols in let (t2, _) = expr (List.hd
tl) symbols in
                          if t1 = t2 then all_match tl else raise (Failure ("Data
Mismatch in MatrixLit: " ^ string_of_data_type t1 ^ " does not match " ^
string_of_data_type t2))
                        else ignore()
          in
          all_match l;
          let expr_mapper e = expr e symbols in
          if List.length d > 2 then (Matrix, SMatrixLit( (List.map expr_mapper l ),
List.hd d, List.hd (List.tl d) ) )
          else if List.length d = 2 then (Matrix, SMatrixLit ( (List.map expr_mapper
(flatten (List.tl d) l)), List.hd d, List.hd (List.tl d)))
          else if List.length d = 1 then (Matrix, SMatrixLit ( (List.map expr_mapper
(flatten (List.tl d) l)), List.hd d, 1))
          else (Matrix, SMatrixLit ( (List.map expr_mapper l ), 0,0))

        | MatrixAccess(s,e1,e2)->
          (Matrix, SMatrixAccess(s, expr e1 symbols, expr e2 symbols))

        | MatrixAccess1D(s,e1)->
          (Matrix, SMatrixAccess1D(s, expr e1 symbols))
        | MatrixAccessCol(s,e1)->
          (Matrix, SMatrixAccessCol(s, expr e1 symbols))
        | MatrixDiagonal(s)->
          (Matrix, SMatrixDiagonal(s))
        | MatrixPower(s,e1)->
          (Matrix, SMatrixPower(s, expr e1 symbols))

        | Id s       -> (type_of_identifier s symbols, SId s)
        | Assign(var, e) as ex ->
            let lt = type_of_identifier var symbols
            and (rt, e') = expr e symbols in
            let err = "illegal assignment " ^ string_of_data_type lt ^ " = " ^
              string_of_data_type rt ^ " in " ^ string_of_expr ex
            in (check_assign lt rt err, SAssign(var, (rt, e')))
        | Unop(op, e) as ex ->
            let (t, e') = expr e symbols in
            let ty = match op with
              Neg when t = Int || t = Double -> t
```

```
          | Not when t = Bool -> Bool
          | _ -> raise (Failure ("illegal unary operator " ^
                                 string_of_uop op ^ string_of_data_type t ^
                                 " in " ^ string_of_expr ex))
          in (ty, SUnop(op, (t, e')))
      | Binop(e1, op, e2) as e ->
          let (t1, e1') = expr e1 symbols
          and (t2, e2') = expr e2 symbols in
          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 = Double -> Double
          | Equal | Neq            when same               -> Bool
          | Less | Leq | Greater | Geq
                       when same && (t1 = Int || t1 = Double) -> Bool
          | And | Or when same && t1 = Bool -> Bool
          | Add | Sub  | Mult when same && t1 = Matrix -> Matrix
          | Mult when t1 = Int && t2 = Matrix -> Matrix
          | Mult when t1 = Double && t2 = Matrix -> Matrix
          | Dot when same && t1 = Matrix -> Double
          | Div when t1 = Matrix && t2 = Int -> Matrix
          | Div when t1 = Matrix && t2 = Double -> Matrix
          | _ -> raise (
        Failure ("illegal binary operator " ^
                      string_of_data_type t1 ^ " " ^ string_of_operator op ^ " " ^
                      string_of_data_type 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 symbols in
            let err = "illegal argument found " ^ string_of_data_type et ^
              " expected " ^ string_of_data_type 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.data_type, SCall(fname, args'))
    in

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

```
                string_of_data_type func.data_type ^ " 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 e symbols = match e with
                [Return _ as s] -> [check_stmt s symbols]
              | Return _ :: _   -> raise (Failure "nothing may follow a return")
              | Block sl :: ss  -> check_stmt_list (sl @ ss) symbols (* Flatten blocks
*)
              | s :: ss         -> let (a,b) = check_stmt s symbols in (a,b) ::
(check_stmt_list ss b)
              | []              -> []
          in let symbols_up = StringMap.empty :: symbols in (SBlock(List.map fst
(check_stmt_list sl symbols_up)), symbols)
        | Vdecl (b, e) ->
          let (te', _) = expr e symbols in
            if (fst b) != te' && te' != Void then raise (Failure ("illegal assignment
" ^ string_of_data_type (fst b)  ^ " = " ^
            string_of_data_type te'))
          else
            let entry = StringMap.add (snd b) (fst b) (List.hd symbols) in let
symbols_up =
            match symbols with
              [_] -> [entry]
            | _ :: tl -> entry :: tl
            | _ -> raise (Failure ("wrong symbol table")) in
                    let k = (SVdecl ((type_of_identifier (snd b) symbols_up, snd b),
expr e symbols_up), symbols_up)
                      in k

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

10.9 codegen.ml

```
(*
Authors: Matcat Team
*)

module L = Llvm
module A = Ast
module S = Sast
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 "matrixLibrary.bc" in
  let llm = Llvm_bitreader.parse_bitcode context llmem in
```

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

  (* Get types from the context *)
  let i32_t      = L.i32_type    context
  and i8_t       = L.i8_type     context
  and i1_t       = L.i1_type     context
  and double_t   = L.double_type context
  and void_t     = L.void_type   context
  and matrx_t    = L.pointer_type (match L.type_by_name llm "struct.matrix" with
      None -> raise (Failure "Missing implementation for struct Matrix")
    | Some t -> t)
  in
  (* Return the LLVM type for a Matcat type *)
  let ltype_of_typ = function
    | A.String -> L.pointer_type (L.array_type i8_t 100)
    | A.Void  -> void_t
    | A.Int   -> i32_t
    | A.Bool  -> i1_t
    | A.Double -> double_t
    | A.Matrix -> matrx_t
  in

  let g_var_suffix =  "" in
  let ip_t       = L.pointer_type i8_t in
  let init t = match t with
          | A.String -> L.const_pointer_null ip_t
          | _      -> L.const_int (ltype_of_typ t) 0
  in

  let global_vars =
  let global_var m (t, n) = StringMap.add n (L.define_global (n ^ g_var_suffix) (init
t) 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 printMatrix_t = L.function_type i32_t [| matrx_t |] in
      let printMatrix_f = L.declare_function "printMatrix" printMatrix_t the_module in

      let matrix_init_t = L.function_type matrx_t [|i32_t ; i32_t|] in
      let matrix_init_f = L.declare_function "initMatrix_helper" matrix_init_t
the_module in

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

      let add_matrix_t = L.function_type matrx_t [|matrx_t; matrx_t|] in
      let add_matrix_f = L.declare_function "matrxAdd" add_matrix_t the_module in

      let sub_matrix_t = L.function_type matrx_t [|matrx_t; matrx_t|] in
      let sub_matrix_f = L.declare_function "matrxSub" sub_matrix_t the_module in

      let transpose_matrix_t = L.function_type matrx_t [|matrx_t|] in
      let transpose_matrix_f = L.declare_function "transpose" transpose_matrix_t
the_module in
```

```
      let mult_matrix_t = L.function_type matrx_t [|matrx_t; matrx_t|] in
      let mult_matrix_f = L.declare_function "matrxMult" mult_matrix_t the_module in

      let det_matrix_t = L.function_type double_t [|matrx_t|] in
      let det_matrix_f = L.declare_function "det" det_matrix_t the_module in

      let inv_matrix_t = L.function_type matrx_t [|matrx_t|] in
      let inv_matrix_f = L.declare_function "inv" inv_matrix_t the_module in

      let dot_matrix_t = L.function_type double_t [|matrx_t; matrx_t|] in
      let dot_matrix_f = L.declare_function "dot" dot_matrix_t the_module in

      let scalar_matrix_t = L.function_type matrx_t [|matrx_t; i32_t|] in
      let scalar_matrix_f = L.declare_function "scaleMatrix" scalar_matrix_t
the_module in

      let scalarDouble_matrix_t = L.function_type matrx_t [|matrx_t; double_t|] in
      let scalarDouble_matrix_f = L.declare_function "scaleMatrixDouble"
scalarDouble_matrix_t the_module in

      let scalarDiv_matrix_t = L.function_type matrx_t [|matrx_t; i32_t|] in
      let scalarDiv_matrix_f = L.declare_function "scalarDivMatrix" scalarDiv_matrix_t
the_module in

      let scalarDivDouble_matrix_t = L.function_type matrx_t [|matrx_t; double_t|] in
      let scalarDivDouble_matrix_f = L.declare_function "scalarDivDoubleMatrix"
scalarDivDouble_matrix_t the_module in

      let rref_matrix_t = L.function_type matrx_t [|matrx_t|] in
      let rref_matrix_f = L.declare_function "rref" rref_matrix_t the_module in

      let isInvertible_matrix_t = L.function_type i1_t [|matrx_t|] in
      let isInvertible_matrix_f = L.declare_function "isInv" isInvertible_matrix_t
the_module in

      let access_matrix_t= L.function_type matrx_t[|matrx_t; i32_t; i32_t|] in
      let access_matrix_f= L.declare_function "accessMatrix" access_matrix_t
the_module in

      let access_matrix1d_t= L.function_type matrx_t[|matrx_t; i32_t|] in
      let access_matrix1d_f= L.declare_function "accessMatrix1D" access_matrix1d_t
the_module in

      let access_matrixcol_t= L.function_type matrx_t[|matrx_t; i32_t|] in
      let access_matrixcol_f= L.declare_function "accessMatrixCol" access_matrixcol_t
the_module in

      let access_matrixdiagonal_t= L.function_type matrx_t[|matrx_t|] in
      let access_matrixdiagonal_f= L.declare_function "get_diagonal"
access_matrixdiagonal_t the_module in

      let power_matrix_t= L.function_type matrx_t[|matrx_t; i32_t|] in
      let power_matrix_f= L.declare_function "power_matrix" power_matrix_t the_module
in

      let rotate90_matrix_t= L.function_type matrx_t[|matrx_t|] in
      let rotate90_matrix_f= L.declare_function "rotate90" rotate90_matrix_t
the_module in

      let symmetry_matrix_t= L.function_type i32_t [|matrx_t|] in
      let symmetry_matrix_f= L.declare_function "check_symmetry" symmetry_matrix_t
the_module in
```

```
  (* let to_imp str = raise (Failure ("To be implemented: " ^ str)) in *)

  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)
      (* Returning mutiple values  *)
      in let ftype = L.function_type (ltype_of_typ fdecl.sdata_type) formal_types in
      StringMap.add name (L.define_function name ftype the_module, fdecl) m in
    List.fold_left function_decl StringMap.empty functions in

  (* 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 float_format_str = L.build_global_stringptr "%g\n" "fmt" builder
    and string_format_str = L.build_global_stringptr "%s\n" "fmt" builder

    in
    let add_local (m, tmp_builder) (t, n) =
      let local_var = L.build_alloca (ltype_of_typ t) n tmp_builder
      in (StringMap.add n local_var m, tmp_builder)
    in
    let local_vars =
      let add_formal m (t, n) p =
        L.set_value_name n p;
       let local = L.build_alloca (ltype_of_typ t) n builder in
        ignore (L.build_store p local builder);
       StringMap.add n local 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, builder) fdecl.slocals
    in

    let scoped_vars = [fst local_vars; global_vars] in

    let rec lookup n var_list = match var_list with
                          hd::tl -> (try StringMap.find n hd with Not_found -> lookup
n tl)
                        | _ -> raise (Failure("wrong variable tables"))
    in


    let rec expr builder ((tp, e) : sexpr) symbol_table = match e with
          SIntLit i  -> L.const_int i32_t i
      | SBoolLit b  -> L.const_int i1_t (if b then 1 else 0)
      | SDoubleLit l -> L.const_float_of_string double_t l
      | SStringLit s -> L.build_global_stringptr s "tmp" builder
      | SId s -> L.build_load (lookup s symbol_table) s builder
      | SMatrixLit (contents, rows, cols) ->
        let rec expr_list = function
          [] -> []
          | hd::tl -> expr builder hd symbol_table::expr_list tl
        in
        let contents' = expr_list contents
        in
```

```
        let m = L.build_call matrix_init_f [| L.const_int i32_t cols; L.const_int
i32_t rows |] "matrix_init" builder
        in
        ignore(List.map (fun v -> L.build_call store_matrix_f [| m ; v |]
"storeEntries" builder) contents'); m
      | SNoexpr     -> L.const_int i32_t 0
      | SNoassign   -> init tp
      | SMatrixAccess(s, e1,e2)->

        let e1'=expr builder e1 symbol_table
        and e2'=expr builder e2 symbol_table
        and matrxPtr = L.build_load (lookup s symbol_table) s builder in
        L.build_call access_matrix_f [|matrxPtr; e1'; e2'|] "accessMatrix" builder

      | SMatrixAccess1D(s, e1)->
          let e1'=expr builder e1 symbol_table
          and matrxPtr = L.build_load (lookup s symbol_table) s builder in
          L.build_call access_matrix1d_f [|matrxPtr; e1'|] "accessMatrix1D" builder

      | SMatrixPower(s, e1)->
        let e1'=expr builder e1 symbol_table
        and matrxPtr = L.build_load (lookup s symbol_table) s builder in
        L.build_call power_matrix_f [|matrxPtr; e1'|] "power_matrix" builder

      | SMatrixAccessCol(s, e1)->
        let e1'=expr builder e1 symbol_table
        and matrxPtr = L.build_load (lookup s symbol_table) s builder in
        L.build_call access_matrixcol_f [|matrxPtr; e1'|] "accessMatrixCol" builder

      | SMatrixDiagonal(s)->
        let matrxPtr = L.build_load (lookup s symbol_table) s builder in
        L.build_call access_matrixdiagonal_f [| (matrxPtr) |] "get_diagonal" builder


      | SAssign (s, e) -> let e' = expr builder e symbol_table in
                          ignore(L.build_store e' (lookup s symbol_table) builder); e'
      | SBinop ((A.Matrix, _) as e1, op, e2)  when op = A.Add || op = A.Sub  || op =
A.Dot || op = A.Mult ->
          let e1' = expr builder e1 symbol_table
          and e2' = expr builder e2 symbol_table in
          (match op with
            A.Add  -> L.build_call add_matrix_f [| e1'; e2' |] "matrxAdd" builder
          | A.Sub  -> L.build_call sub_matrix_f [| e1'; e2' |] "matrxSub" builder
          | A.Mult -> L.build_call mult_matrix_f [| e1'; e2' |] "matrxMult" builder
          | A.Dot -> L.build_call dot_matrix_f [| e1'; e2' |] "dot" builder
          | _ -> raise (Failure "not implemented")
          )

       | SBinop (((A.Int,_)  as e1), op, ((A.Matrix,_) as e2))  when  op = A.Mult ->
          let e1' = expr builder e1 symbol_table
          and e2' = expr builder e2 symbol_table in
          (match op with
          | A.Mult -> L.build_call scalar_matrix_f [| e2'; e1' |] "scaleMatrix"
builder
          | _ -> raise (Failure "not implemented")
          )

         | SBinop (((A.Double,_)  as e1), op, ((A.Matrix,_) as e2))  when  op = A.Mult
->
          let e1' = expr builder e1 symbol_table
          and e2' = expr builder e2 symbol_table in
          (match op with
```

```
          | A.Mult -> L.build_call scalarDouble_matrix_f [| e2'; e1' |]
"scaleMatrixDouble" builder
          | _ -> raise (Failure "not implemented")
          )

       | SBinop (((A.Matrix,_)  as e1), op, ((A.Int,_) as e2))  when op = A.Div ->
          let e1' = expr builder e1 symbol_table
          and e2' = expr builder e2 symbol_table in
          (match op with
          | A.Div -> L.build_call scalarDiv_matrix_f [| e1'; e2' |] "scalarDivMatrix"
builder
          | _ -> raise (Failure "not implemented")
          )

          | SBinop (((A.Matrix,_)  as e1), op, ((A.Double,_) as e2))  when op = A.Div
->
          let e1' = expr builder e1 symbol_table
          and e2' = expr builder e2 symbol_table in
          (match op with
          | A.Div -> L.build_call scalarDivDouble_matrix_f [| e1'; e2' |]
"scalarDivDoubleMatrix" builder
          | _ -> raise (Failure "not implemented")
          )

      | SBinop ((A.Double,_ ) as e1, op, e2) ->
        let e1' = expr builder e1 symbol_table
        and e2' = expr builder e2 symbol_table 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.Cr      -> raise (Failure "cr can be used only for matrices")
        | A.Dot     -> raise (Failure "cr can be used only for matrices")
        | A.And | A.Or ->
            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 symbol_table
        and e2' = expr builder e2 symbol_table in
        (match op with
          A.Add     -> L.build_add
        | A.Sub     -> L.build_sub
        | A.Mult    -> L.build_mul
        | A.Div     -> L.build_sdiv
        | A.And     -> L.build_and
        | A.Or      -> L.build_or
        | A.Equal   -> L.build_icmp L.Icmp.Eq
        | A.Neq     -> L.build_icmp L.Icmp.Ne
        | A.Less    -> L.build_icmp L.Icmp.Slt
        | A.Leq     -> L.build_icmp L.Icmp.Sle
        | A.Greater -> L.build_icmp L.Icmp.Sgt
        | A.Geq     -> L.build_icmp L.Icmp.Sge
        | A.Cr      -> raise (Failure "cr can be used only for matrices")
        | A.Dot     -> raise (Failure "cr can be used only for matrices")
         ) e1' e2' "tmp" builder
```

```
        | SUnop(op, ((t, _) as e)) ->
          let e' = expr builder e symbol_table in
               (match op with
               A.Neg when t = A.Double -> L.build_fneg
             | A.Neg                  -> L.build_neg
        | A.Not                  -> L.build_not) e' "tmp" builder
        | SCall ("print", [e]) | SCall ("printb", [e]) ->
          L.build_call printf_func [| int_format_str ; (expr builder e symbol_table)
|]
          "printf" builder

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

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

        | SCall ("printm", [e]) ->
          L.build_call printMatrix_f [| (expr builder e symbol_table) |] "printm"
builder

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

        | SCall ("inv", [e]) ->
        L.build_call inv_matrix_f [| (expr builder e symbol_table) |] "inv" builder

        | SCall ("det", [e]) ->
         L.build_call det_matrix_f [| (expr builder e symbol_table) |] "det" builder

        | SCall ("rref", [e]) ->
         L.build_call rref_matrix_f [| (expr builder e symbol_table) |] "rref" builder

        | SCall ("isInv", [e]) ->
         L.build_call isInvertible_matrix_f [| (expr builder e symbol_table) |]
"isInv" builder

        | SCall ("get_diagonal", [e]) ->
         L.build_call access_matrixdiagonal_f [| (expr builder e symbol_table) |]
"get_diagonal" builder

        | SCall ("rotate90", [e]) ->
          L.build_call rotate90_matrix_f [| (expr builder e symbol_table) |]
"rotate90" builder

        | SCall ("check_symmetry", [e]) ->
          L.build_call symmetry_matrix_f [| (expr builder e symbol_table) |]
"check_symmetry" builder


        | SCall (f, args) ->
          let (fdef, fdecl) = StringMap.find f function_decls in
      let llargs_helper b st x = expr b x st in
           let llargs = List.rev (List.map (llargs_helper builder symbol_table)
(List.rev args)) in
           let result = (match fdecl.sdata_type with
                        A.Void -> ""
```

```
                    | _ -> f ^ "_result") in
      L.build_call fdef (Array.of_list llargs) result builder
    in
    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, symbol_table, pbuilder) = function
             SBlock sl -> List.fold_left stmt (builder, StringMap.empty ::
symbol_table, pbuilder) sl
           | SExpr e -> let _ = expr builder e symbol_table in (builder, symbol_table,
pbuilder)
           | SVdecl (e1, e2) ->
              let (tp, s) =
              e1 in let (symbol_table2, _) =
                add_local (List.hd symbol_table, pbuilder) (tp, s) in
                  let symbol_table3 = (symbol_table2 :: (List.tl symbol_table)) in

                  let _ = match e2 with
                    (A.Void, SNoassign) -> init tp
                  | _ -> expr builder (tp, SAssign(s, e2)) symbol_table3
           in (builder, symbol_table3, pbuilder)
           | SReturn e -> let _ = match fdecl.sdata_type with
                                 (* A.Int -> L.build_ret (expr builder e symbol_table)
builder
                               | A.Double -> L.build_ret (expr builder e symbol_table)
builder
                               | A.Matrix -> L.build_ret (expr builder e symbol_table)
builder *)
                               | _ -> L.build_ret (expr builder e symbol_table) builder
                     in (builder, symbol_table, pbuilder)
           | SIf (predicate, then_stmt, else_stmt) ->
              let bool_val = expr builder predicate symbol_table 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
                    let (b', _, _) = (stmt ((L.builder_at_end context then_bb),
symbol_table, pbuilder) then_stmt) in
                let () = add_terminal b' build_br_merge in

        let else_bb = L.append_block context "else" the_function in
        let (b', _, _) = (stmt ((L.builder_at_end context else_bb), symbol_table,
pbuilder) else_stmt) in
        let () = add_terminal b' build_br_merge in

        let _ = L.build_cond_br bool_val then_bb else_bb builder in
        (L.builder_at_end context merge_bb, symbol_table, 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
         let (b', _, _) = (stmt ((L.builder_at_end context body_bb), symbol_table,
pbuilder) body) in
         add_terminal b'
           (L.build_br pred_bb);
```

```
          let pred_builder = L.builder_at_end context pred_bb in
          let bool_val = expr pred_builder predicate symbol_table 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, symbol_table, L.builder_at_end context
merge_bb)



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

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

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

        List.iter build_function_body functions;
        the_module
```

10.10 matrixLibrary.c

```
/*
Authors:
Davit, Mariam, Andreas (minor fixes only)
*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <unistd.h>


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

struct matrix {
  int num_rows;
  int num_cols;
  float** matrixAddr; // accessed [row][col]
  int buildPosition;
};
typedef struct matrix matrix;
matrix* inverseHelper(matrix*, matrix*, float);
double determinant(matrix*, int);
double inverse(matrix*, int);
matrix* cofactor(matrix*, float);
double det(matrix*);
```

```
matrix* rref_helper(matrix*, double);
matrix* accessMatrixDgl(matrix*);
matrix* power_matrix_helper(matrix*, int);

int debug = 0;


void reverseMatrix(matrix* input) {
    int row = input->num_rows;
    int col = input->num_cols;

    for(int i = 0; i<row; i++) {
        for(int j=0; j<col/2; j++) {
          float temp = input->matrixAddr[i][j];
          input->matrixAddr[i][j] = input->matrixAddr[i][col-1-j];
          input->matrixAddr[i][col-1-j] = temp;

        }
    }
}
void printMatrix(matrix* input) {
    int row = input->num_rows;
    int col = input->num_cols;
    int temp;
    printf("(");
    printf("\n");
    for(int i = 0; i<row; i++) {
       printf("[");
        for(int j=0; j < col; j++) {
            printf("%.2f ", input->matrixAddr[i][j]);
        }
        printf("]");
        printf("\n");
    }
    printf(")");
    printf("\n");

}




matrix* initMatrix(float* listOfValues, int num_cols, int num_rows) {
  float** matrixValues = malloc(num_rows * sizeof(float*));

  if(debug == 1) {
      printf("Building matrix:\n");
      printf("num_rows: %d\n", num_rows);
      printf("num_cols: %d\n", num_cols);
  }

  //set all values in matrix to NULL if list of values is NULL
  if (listOfValues == NULL) {
    for(int i = 0; i < num_rows; i++) {
      float* matrix_row = malloc(num_cols * sizeof(float));
      *(matrixValues + i) = matrix_row;
      for(int j = 0; j < num_cols; j++) {
        matrix_row[j] = 0;
      }
    }
  }

  //load values from a list of values
```

```
  else {
    for(int i = 0; i < num_cols; i++) {
      float* matrix_col = malloc(num_rows * sizeof(float));
      *(matrixValues + i) = matrix_col;
      for(int j = 0; j < num_rows; j++) {
        matrix_col[j] = listOfValues[i*num_rows + j];
      }
    }
  }

  //return a pointer to matrix struct
  matrix* result = malloc(sizeof(struct matrix));
  result->num_cols = num_cols;
  result->num_rows = num_rows;
  result->matrixAddr = matrixValues;
  result->buildPosition = 0;
  return result;
}



matrix* initMatrix_helper( int num_cols, int num_rows) {
    return initMatrix(NULL, num_cols, num_rows);
}

matrix* storeEntries(matrix* target, int value) {
    int position = target->buildPosition;
    int curr_row = position / target->num_cols;
    int curr_col = position % target->num_cols;

    if(debug == 1) {
        printf("Storing: %d\n", value);
        printf("in row: %d\n", curr_row);
        printf("in col: %d\n", curr_col);
    }

    target->matrixAddr [curr_row][curr_col] = value;
    target->buildPosition = target->buildPosition + 1;
    return target;
}

int has_same_dim(matrix* lhs, matrix* rhs){
  int same_num_rows = lhs->num_rows == rhs->num_rows;
  int same_num_cols = lhs->num_cols == rhs->num_cols;
  return same_num_rows && same_num_cols;
}

matrix* matrxAdd(matrix* lhs, matrix* rhs) {
  //check dimensions
  if (!has_same_dim(lhs, rhs)) {
    die("matrix add size mismatch");
  }
  int rows = lhs->num_rows;
  int cols= lhs->num_cols;
  matrix *result = initMatrix(NULL, cols, rows);
  for(int i=0; i<rows; i++) {
    for(int j=0; j<cols; j++) {
        int sum = lhs->matrixAddr[i][j] + rhs->matrixAddr[i][j];
        result->matrixAddr[i][j] = sum;
    }
  }
```

```
  return result;
}

matrix* matrxSub(matrix* lhs, matrix* rhs) {
  //check dimensions
  if (!has_same_dim(lhs, rhs)) {
    die("matrix sub size mismatch");
  }
  int rows = lhs->num_rows;
  int cols= lhs->num_cols;
  matrix *result = initMatrix(NULL, cols, rows);
  for(int i=0; i<rows; i++) {
    for(int j=0; j<cols; j++) {
        int sum = lhs->matrixAddr[i][j] - rhs->matrixAddr[i][j];
        result->matrixAddr[i][j] = sum;
    }
  }

  return result;
}


matrix* transpose(matrix* input) {
    int rows = input->num_cols;
    int cols = input->num_rows;

    float** matrixValues = malloc(cols * sizeof(float*));

    for (int i = 0; i < rows; i++) {
        float* matrix_col = malloc(rows * sizeof(float));
        *(matrixValues + i) = matrix_col;
        for (int j = 0; j < cols; j++) {
            matrix_col[j] = *(*((input->matrixAddr) + j)+i);
        }
    }
    input->num_rows = rows;
    input->num_cols = cols;
    input->matrixAddr = matrixValues;
    return input;
}



matrix* matrxMult(matrix* lhs, matrix* rhs) {
  //check dimensions
  if (lhs->num_rows != rhs->num_rows || lhs->num_cols != rhs->num_rows) {
    die("matrix mult size mismatch");
  }

  matrix* originalMatrix =initMatrix(NULL,lhs->num_cols,lhs->num_rows);

  for(int i = 0; i < lhs->num_rows;i++){
      for(int j = 0; j < lhs->num_cols; j++){
        originalMatrix->matrixAddr[i][j] =lhs->matrixAddr[i][j];
           }
       }
  int rows_lhs = lhs->num_rows;
  int cols_lhs = lhs->num_cols;
  int rows_rhs = rhs->num_rows;
  int cols_rhs = rhs->num_cols;

  matrix *result = initMatrix(NULL, rows_lhs, cols_lhs);
  for(int i=0; i<rows_lhs; i++) {
```

```
    for(int j=0; j<cols_rhs; j++) {
      for (int k = 0; k < cols_rhs; k++){
        result->matrixAddr[i][j] = result->matrixAddr[i][j] + (originalMatrix-
>matrixAddr[i][k] * rhs->matrixAddr[k][j]);
      }
    }
  }

  return result;
}

matrix* inv(matrix* input){
  int rows = input->num_rows;
  int cols = input->num_cols;
  double d = determinant(input,rows);
  if(d == 0){
    die("\nInverse of the matrix is not possible\n");
  }
   input = (cofactor(input, rows));
   return input;

}

matrix* cofactor(matrix* input, float f)
{
 matrix *b = initMatrix(NULL, (int)f,(int)f);
 matrix *fac = initMatrix(NULL,(int) f,(int)f);

 int p, q, m, n, i, j;
 for (q = 0;q < f; q++)
 {
   for (p = 0;p < f; p++)
    {
     m = 0;
     n = 0;
     for (i = 0;i < f; i++)
     {
       for (j = 0;j < f; j++)
        {
          if (i != q && j != p)
          {
            b->matrixAddr[m][n] = input->matrixAddr[i][j];
            if (n < (f - 2))
             n++;
            else
             {
               n = 0;
               m++;
               }
            }
        }
      }
      fac->matrixAddr[q][p] = pow(-1, q + p) * determinant(b, f - 1);
    }
  }
  return inverseHelper(input,fac,f);

}

matrix* inverseHelper(matrix* input, matrix* fac, float r)
{
  int i, j;
  float d;
```

```
  matrix *inverse = initMatrix(NULL,(int)r ,(int) r);
  matrix *b = initMatrix(NULL,(int)r ,(int) r);

  for (i = 0;i < r; i++)
    {
     for (j = 0;j < r; j++)
       {
         b->matrixAddr[i][j] = fac->matrixAddr[j][i];
       }
    }

  d = determinant(input,r);
  for (i = 0;i < r; i++)
    {
     for (j = 0;j < r; j++)
       {
        inverse->matrixAddr[i][j] = b->matrixAddr[i][j] / d;
        }
    }
   return inverse;
}

double det(matrix* input) {

  int rows = input->num_rows;
  int cols = input->num_cols;
  if(rows != cols){
    die("Finding a determinant of non-square matrix is not possible in our universe");
  }

  double deter = determinant(input, cols);
  return deter;


}


// source code: https://www.sanfoundry.com/c-program-find-inverse-matrix/
double determinant(matrix* input, int k)
{
float s = 1;
double det = 0;
int rows = input->num_rows;
int cols = input->num_cols;
matrix *result = initMatrix(NULL,cols , rows);
int i, j, m, n, c;
if (k == 1)
  {
   return (input->matrixAddr[0][0]);
  }
  else
    {
     det = 0;
     for (c = 0; c < k; c++)
       {
        m = 0;
        n = 0;
        for (i = 0;i < k; i++)
          {
            for (j = 0 ;j < k; j++)
              {
                result->matrixAddr[i][j] = 0;
```

```
                if (i != 0 && j != c)
                 {
                   result->matrixAddr[m][n] = input->matrixAddr[i][j];
                   if (n < (k - 2))
                    n++;
                   else
                    {
                     n = 0;
                     m++;
                     }
                   }
                }
              }

          det = det + s * (input->matrixAddr[0][c] * determinant(result, k - 1));
          s = -1 * s;
          }
    }

    return det;
}

double dot(matrix* lhs, matrix* rhs) {
    //check to make sure matrices are the same size
    if (lhs->num_rows != rhs->num_rows || lhs->num_cols != rhs->num_cols) {
        die("Matrices are not the same size!");
    }
    double dotPr = 0.0;
    //once we know that matrices are same size, we can compute result
    for (int i=0; i < lhs->num_rows; i++)
    {
        for (int j=0; j < rhs->num_cols; j++)
        {
          dotPr += lhs->matrixAddr[i][j] * rhs->matrixAddr[i][j];
        }
    }
    return dotPr;
}


matrix* scaleMatrix(matrix* input, int scalar) {
  int rows = input->num_rows;
  int cols= input->num_cols;
  matrix *result = initMatrix(NULL, cols, rows);

  for(int i=0; i<rows; i++) {
    for(int j=0; j < cols; j++) {
        double product = input->matrixAddr[i][j] * scalar;
        result->matrixAddr[i][j] = product;
    }
  }

  return result;
}

matrix* scaleMatrixDouble(matrix* input, double scalar) {
  int rows = input->num_rows;
  int cols= input->num_cols;
  matrix *result = initMatrix(NULL, cols, rows);


  for(int i=0; i<rows; i++) {
    for(int j=0; j < cols; j++) {
```

```
        double product = input->matrixAddr[i][j] * scalar;
        result->matrixAddr[i][j] = product;
    }
  }

  return result;
}

matrix* scalarDivMatrix(matrix* input, int scalar) {
  int rows = input->num_rows;
  int cols= input->num_cols;
  matrix *result = initMatrix(NULL, cols, rows);

  for(int i=0; i<rows; i++) {
    for(int j=0; j < cols; j++) {
        float product = input->matrixAddr[i][j] / scalar;
        result->matrixAddr[i][j] = product;
    }
  }
  return result;
}

matrix* scalarDivDoubleMatrix(matrix* input, double scalar) {
  if(scalar == 0){
    die("division by 0 is not allowed in thsi universe");
  }
  int rows = input->num_rows;
  int cols= input->num_cols;
  matrix *result = initMatrix(NULL, cols, rows);


  for(int i=0; i<rows; i++) {
    for(int j=0; j < cols; j++) {
        double product = input->matrixAddr[i][j] / scalar;
        result->matrixAddr[i][j] = product;
    }
  }
  return result;
}


void cleanUpMatrix (matrix* oMatrix,  double tolerance) {

       // Removes all numbers close to zero, i.e between -tol and +tol
       for (int i = 0; i < oMatrix->num_rows; i++)
              for (int j = 0; j < oMatrix->num_cols; j++)
                     if (fabs (oMatrix->matrixAddr[i][j]) < tolerance)
                            oMatrix->matrixAddr[i][j] = 0;
}

void exchangeRows (matrix* oMatrix, int r1, int r2) {

       double t = 0;
       for (int i = 0; i < oMatrix->num_cols; i++) {
              t = oMatrix->matrixAddr[r1][i];
              oMatrix->matrixAddr[r1][i] = oMatrix->matrixAddr[r2][i];
              oMatrix->matrixAddr[r2][i] = t;
       }
}

double getValue (matrix* oMatrix, int i, int j) {

       if ((i < 0) || (j < 0)) {
```

```
		printf("Error in indexing\n");
	}

	if ((i >= oMatrix->num_rows) || (j >= oMatrix->num_cols)) {
		printf ("Error in indexing: %d, %d\n", i, j);
    exit(0);
	}

	return oMatrix->matrixAddr[i][j];
}



matrix* rref_helper(matrix* oMatrix, double tolerance)
{
	int currentRow; double factor;

	matrix* oEchelon = initMatrix(NULL, oMatrix->num_cols,oMatrix->num_rows);

	// Make a copy and work on that.
	for (int i = 0; i < oMatrix->num_rows; i++)
		for (int j = 0; j < oMatrix->num_cols; j++)
			oEchelon->matrixAddr[i][j] = oMatrix->matrixAddr[i][j];

	int Arow = 0; int Acol = 0;
	while ((Arow < oEchelon->num_rows) && (Acol < oEchelon->num_cols)) {
		// locate a nonzero column
		if (abs (getValue (oEchelon, Arow, Acol) < tolerance)) {
			// If the entry is zero work our way down the matrix
			// looking for a nonzero entry, when found, swap it for Arow
			currentRow = Arow;
			do {

				// next row
				currentRow++;
				// Have we reached the end of the rows but we've still got
columns left to scan?
				if ((currentRow >= oEchelon->num_rows) && (Acol <=
oEchelon->num_cols)) {
					// reset row counter back to where it was and try
next column
					currentRow = Arow; Acol++;
				}

				// If we've scanned the whole matrix, then lets get out...
				if (currentRow >= oEchelon->num_rows) {
					cleanUpMatrix (oEchelon, tolerance);
					return oEchelon;
				}
			} while (fabs (getValue (oEchelon, currentRow, Acol)) <
tolerance);

			// We've found a nonzero row entry so swap it with 'Arow' which
did have a zero as its entry
			exchangeRows (oEchelon, Arow, currentRow);
		}
		// Arow now holds the row of interest }
		factor = 1.0 / getValue (oEchelon, Arow, Acol);
		// reduce all the entries along the column by the factor
		for (int i = Acol; i < oEchelon->num_cols; i++){
			oEchelon->matrixAddr[Arow][i] = getValue (oEchelon, Arow, i) *
factor;
```

```
    }

              // now eliminate all entries above and below Arow, this generates the
reduced form
              for (int i = 0; i < oEchelon->num_rows; i++) {
                     // miss out Arow itself
                     if ((i != Arow) && (fabs (getValue (oEchelon, i, Acol)) >
tolerance)) {
                            factor = getValue (oEchelon, i, Acol);
                            // work your way along the column doing the same operation
                            for (int j = Acol; j < oEchelon->num_cols; j++) {
                                   oEchelon->matrixAddr[i][j] = getValue (oEchelon, i,
j) - factor * getValue (oEchelon, Arow, j);
                            }
                     }
              }

              Arow++; Acol++;
       }
       cleanUpMatrix (oEchelon, tolerance);
       return oEchelon;
}

matrix* rref(matrix* input) {
  return rref_helper(input,1e-6);

}

void isInv(matrix* input) {
  double d = determinant(input,input->num_rows);
  if (d == 0){
    printf("%s\n","This matrix is not invertible");
  }else{
    printf("%s\n","This matrix is invertible");
  }
}


matrix* accessMatrix(matrix* input, int row, int col){
  if(row >= input->num_rows || col >= input->num_cols){
    die("Matrix index out of bound");
  }
  matrix* result=initMatrix(NULL, 1, 1);
  result->matrixAddr[0][0]=input->matrixAddr[row][col];
  return result;


}

matrix* accessMatrix1D(matrix* input, int row){
  if(row >= input->num_rows){

    die("Matrix index out of bound");
  }

   matrix* result=initMatrix(NULL, input->num_cols, 1);
  for(int i = row; i < row+1;i++){
           for(int j = 0; j < input->num_cols; j++){
             result->matrixAddr[0][j] =input->matrixAddr[i][j];

           }
         }
         return result;
```

```
}

matrix* accessMatrixCol(matrix* input, int col){
  if(col >= input->num_cols){
    die("Matrix index out of bound");
  }

   matrix* result=initMatrix(NULL, input->num_rows, 1);
  for(int i = 0; i < input->num_rows;i++){
           for(int j = col; j < col+1; j++){
             result->matrixAddr[0][i] =input->matrixAddr[i][j];

           }
         }

         return result;

}


matrix* get_diagonal(matrix* input){
  if(input->num_cols != input->num_rows){
    die("entry should be squared matrix");
  }
  matrix* result=initMatrix(NULL, input->num_rows, 1);
  for(int i = 0; i < input->num_rows;i++){
           for(int j = 0; j < input->num_cols; j++){

             if(i == j){
                result->matrixAddr[0][i] =input->matrixAddr[i][j];
             }
           }
         }

         return result;

}

matrix* power_matrix(matrix* input, int power){
  if(input->num_rows == input->num_cols){
      if(power < 0 ){
      matrix* result=initMatrix(NULL, input->num_cols,input->num_rows);
      result = inv(input);
      int newPower = abs(power);
      result = power_matrix_helper(result, newPower);
      return result;

    }
    if( power == 0){
      matrix* identityMatrix =initMatrix(NULL,input->num_cols,input->num_rows);
      for(int i = 0; i < input->num_rows;i++){
            for(int j = 0; j < input->num_cols; j++){
              if(i == j){
                identityMatrix->matrixAddr[i][j] = 1;
              }else{
                identityMatrix->matrixAddr[i][j] = 0;
              }
            }
      }

      return identityMatrix;
    }
```

```
  if(power > 0){
     return power_matrix_helper(input,power);

  }
  }else{
    die("I need squared matrix to continue calculations");
  }
  return input;
}



matrix* power_matrix_helper(matrix* input, int power){
  if(power == 1){
    return input;
  }
  matrix* tempMatrix =initMatrix(NULL,  input->num_cols, input->num_rows);
  matrix* originalMatrix =initMatrix(NULL,  input->num_cols,input->num_rows);

  for(int i = 0; i < input->num_rows;i++){
      for(int j = 0; j < input->num_cols; j++){
        originalMatrix->matrixAddr[i][j] =input->matrixAddr[i][j];
           }
       }

  while(power > 1){
    power--;
    tempMatrix = matrxMult(input, originalMatrix);
    input = tempMatrix;

  }


  return input;

}


void rotate90(matrix* input)
{
  if(input->num_cols != input->num_rows){
    die("I don't like non-squared matrices until my create will add me that feature");
  }
  int rows = input->num_rows;
  int cols = input->num_cols;

    // Traverse each cycle
    for (int i = 0; i < rows / 2; i++) {
        for (int j = i; j < cols - i - 1; j++) {

            // Swap elements of each cycle
            // in clockwise direction
            int temp = input->matrixAddr[i][j];
            input->matrixAddr[i][j] = input->matrixAddr[rows - 1 - j][i];
            input->matrixAddr[rows - 1 - j][i] = input->matrixAddr[rows - 1 - i][rows
- 1 - j];
            input->matrixAddr[rows - 1 - i][rows - 1 - j] = input->matrixAddr[j][rows
- 1 - i];
            input->matrixAddr[j][rows - 1 - i] = temp;
        }
    }
}
int check_symmetry(matrix* input){
```

```
  int isSymmetric;
  matrix* transposeM = initMatrix(NULL,input->num_cols, input->num_rows);
  transposeM = transpose(input);

  isSymmetric = 1;
    for(int row=0; row < input->num_rows && isSymmetric; row++)
    {
        for(int col=0; col < input->num_cols; col++)
        {
            /* If matrix A is not equal to its transpose */
            if(input->matrixAddr[row][col] != transposeM->matrixAddr[row][col])
            {
                isSymmetric = 0;
                break;
            }
        }
    }
    return isSymmetric;
}

#ifdef BUILD_TEST
int main(int argc,char** argv) {
  //run tests of each function
  //initMatrix and display of empty matrix
  matrix *null_matrix=initMatrix(NULL, 2, 2);
  printf("NULL MATRIX: \n");
  display(null_matrix);

  //initMatrix and display of 2x2 matrix
  int vals1[] = {3, 8, 4, 6};
  int *list1 = vals1;
  matrix *m = initMatrix(list1, 2, 2);
  printf("2x2 MATRIX: \n");
  display(m);

  //TODO test codegen builder
  for( int i = 0; i < 4; i++) {
      m = storeEntries(m, 5);
      printf("Stroring 5: \n");
      printMatrix(m);
  }


}
#endif
```

10.11 Makefile

```
# Authors: Matcat Team, modified from MicroC and referenced Shoo
.PHONY : all
all : clean matcat.native matrixLibrary.o

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

matrixLibrary : matrixLibrary.c
       gcc -o matrixLibrary -DBUILD_TEST matrixLibrary.c -lm

matrixLibrary.bc : matrixLibrary.c
```

```
	clang -emit-llvm -o matrixLibrary.bc -c matrixLibrary.c -Wno-varargs

%.cmo : %.ml
	ocamlc -c $<

%.cmi : %.mli
	ocamlc -c $<

parser.ml : parser.mly
	ocamlyacc $^

scanner.ml : scanner.mll
	ocamllex $^

.PHONY : clean
clean :
	ocamlbuild -clean
	rm -rf testall.log ocamlllvm *.diff *.err *.ll *.lli *.exe *.out *.s
matcat.native *.o *.bc

.PHONY : test
test : matcat.native
	./testall.sh

# An excluding way to build the tarball
EXCLUDEPATTERNS = .* *_build* *.log *.pdf
ECSTRING = $(EXCLUDEPATTERNS:%= --exclude="%" )
current_dir = $(notdir $(CURDIR))
tar : clean
	echo ${ECSTRING}
	cd .. && tar ${ECSTRING} -cvf ${current_dir}/matcat.tar.gz ${current_dir}
```

10.12 Dockerfile

```
# MicroC Dockerfile modified by Davit and Andreas
# Based on 18.04 LTS
FROM ubuntu:bionic

RUN apt-get -yq update && \
    apt-get -y upgrade && \
    apt-get -yq --no-install-suggests --no-install-recommends install \
    ocaml \
    menhir \
    llvm-6.0 \
    llvm-6.0-dev \
    m4 \
    git \
    aspcud \
    ca-certificates \
    python2.7 \
    pkg-config \
    cmake \
    clang \
    opam

# For testall.py and autosign.py
RUN apt-get -yq --no-install-suggests --no-install-recommends install -y python3-pip
python3-dev \
    && cd /usr/local/bin \
    && ln -s /usr/bin/python3 python \
    && pip3 install --upgrade pip \
    && pip3 install colorama \
    && pip3 install GitPython

RUN ln -s /usr/bin/lli-6.0 /usr/bin/lli
RUN ln -s /usr/bin/llc-6.0 /usr/bin/llc

RUN opam init
RUN opam install \
    llvm.6.0.0 \
    ocamlfind

WORKDIR /root

ENTRYPOINT ["opam", "config", "exec", "--"]

CMD ["bash"]
```

10.13 autosign.py

```
"""
Author: Andreas
Desc.:  This script sign our testcases
Refs:   https://gitpython.readthedocs.io/en/stable/tutorial.html
Note:   This script is mostly hard-coded
"""

import os
from git import Repo

mcFiles = []
DIR = "tests"
# DIR = "future_improvements"

for root, dirs, files in os.walk(DIR):
    for file in files:
        if file.endswith('.mc'):
            mcFiles.append(os.path.join(root, file))

repo = Repo(".")
assert not repo.bare

for file in mcFiles:
    with open(file, 'r+') as f:
        content = f.read()
        first_line = content.splitlines()[0]
        if first_line.startswith(("/*", "//")):
            # print(f"{file} is signed.")
            continue

        commits = repo.iter_commits('--all', paths=file)
        print(file)
        authors = []
        for commit in commits:
            hexsha = commit.hexsha

            # ignore MicroC creation/removal commits
            if hexsha.startswith(("3b6f80de", "0e29cf4")):
                continue

            author = commit.committer.name.split()[0]
            if author in authors or author.startswith("GitHub"):
                continue
            authors.append(author)
        print(authors)
        if len(authors) == 1:
            f.seek(0, 0)
            line = f"// Author: {authors[0]}"
            f.write(line.rstrip('\r\n') + '\n' + content)
```

10.14 _tags

```
# Author: Based on MicroC by Andreas

# Include the llvm and llvm.analysis packages while compiling
true: package(llvm), package(llvm.analysis)

# Enable almost all compiler warnings
true : warn(+a-4)

# Instruct ocamlbuild to ignore the "printbig.o" file when it's building
"matrixLibrary.o": not_hygienic
```

10.15 run.sh

```
#!/bin/sh
# Author:           Andreas
# Usage:            ./mc.sh <filename>
# Description:      This shell script compiles and runs a single .mc file to test a
single test case faster
#                          It doesn't check. It simply runs.

LLI="lli"
LLC="llc"
CC="cc"
MATCAT="./matcat.native"
ulimit -t 5

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

Run() {
    echo $* 1>&2
    eval $* || {
        SignalError "$1 failed on $*"
        return 1
    }
}

basename=`echo $1 | sed 's/.*\\///
                         s/.mc//'`
reffile=`echo $1 | sed 's/.mc$//'`
basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

Run $MATCAT $1 >$basename.ll &&
Run "$LLC" "-relocation-model=pic" "${basename}.ll" &&
Run "$CC" "-o" "${basename}.exe" "${basename}.s" "matrixLibrary.o" "-lm"&&
Run ./$basename.exe

rm -f "$basename.exe"
rm -f "$basename.s"
rm -f "$basename.ll"
```

10.16 createTest.sh

```
#!/bin/bash
# Author: James
# Automate making test files and corresponding .out/.err files


echo 'func main() int {
    matrix m;
    m = [[1,2,3]];
    printm(m);

    return 0;
}' > tests/$1.mc

if [[ "$1" == *"test-"* ]]; then
    touch tests/$1.out
fi


if [[ "$1" == *"fail-"* ]]; then
    touch tests/$1.err
fi
```

10.17 testall.py

```
"""
Author:         Andreas
Description:    Replace testall.sh
Requirements:   Python >3.6, and the imported libraries
Usage:
                py testall.py <test_dir, default: tests>
                Check if matcat.native is built
                Check if there is any weird file in ./tests
                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
References:
                https://docs.python.org/3/library/argparse.html
                https://docs.python.org/3/library/fnmatch.html#fnmatch.fnmatch
                https://stackoverflow.com/questions/3207219/how-do-i-list-all-files-
of-a-directory
                https://stackoverflow.com/questions/678236/how-to-get-the-filename-
without-the-extension-from-a-path-in-python
                https://stackoverflow.com/questions/3751900/create-file-path-from-
variables
                https://stackoverflow.com/questions/54208417/how-to-check-if-popen-
from-subprocess-throws-an-error
                https://stackoverflow.com/questions/11540854/file-as-command-line-
argument-for-argparse-error-message-if-argument-is-not-va
                https://stackoverflow.com/questions/38834378/path-to-a-directory-as-
argparse-argument
"""
import argparse
from colorama import Fore, Style
import fnmatch
import os
import subprocess


LLI = "lli"
```

```
LLC = "llc"
CC = "cc"
MCEXTENSION = ".mc"
OUT = ".out"
ERR = ".err"
MATCAT = "matcat.native"
TESTS_DIR_NAME = "tests"
RUNSH = "./run.sh"


def print_w(s):
    print(Fore.RED + f"WARNING: {s}")
    print(Style.RESET_ALL, end='')


def print_r(s):
    print(Fore.RED + s)
    print(Style.RESET_ALL, end='')


def print_g(s):
    print(Fore.GREEN + s)
    print(Style.RESET_ALL, end='')

# https://stackoverflow.com/a/51212150/13109740
def dir_path(string):
    if os.path.isdir(string):
        return string
    else:
        raise NotADirectoryError(string)


parser = argparse.ArgumentParser(
    description='Fancy Automated Integreation Test for Matcat')
parser.add_argument('--dir',
                    dest="tests_dir", default=TESTS_DIR_NAME,
                    help='select the tests directory', metavar="FILE",
                    type=dir_path)
args = parser.parse_args()

tests_dir = args.tests_dir

if MATCAT not in os.listdir("."):
    print_w(f"{MATCAT} not found. Please make Matcat first.")
    exit(-1)

positive_tests = []
negative_tests = []
expected_res = []
for file in os.listdir(tests_dir):
    if fnmatch.fnmatch(file, f'test*{MCEXTENSION}'):
        positive_tests.append(file)
    elif fnmatch.fnmatch(file, f'fail*{MCEXTENSION}'):
        negative_tests.append(file)
    elif fnmatch.fnmatch(file, f'*{OUT}') or fnmatch.fnmatch(file, f'*{ERR}'):
        expected_res.append(file)
    else:
        print_w(f"{file} is not recognized.")


def get_fname(file):
    base = os.path.basename(file)
    fname = os.path.splitext(base)[0]
```

```
    return fname


def clean_up(fname):
    os.remove(f"./{fname}.ll")
    os.remove(f"./{fname}.s")
    os.remove(f"./{fname}.exe")


# Book keeping vars
checked = 0
skipped = 0
passed = 0
for test in positive_tests + negative_tests:
    is_positive_test = test in positive_tests
    expected_fname = get_fname(test)

    if is_positive_test:
        expected_fname += f"{OUT}"
    else:
        expected_fname += f"{ERR}"
    if expected_fname not in expected_res:
        print_w(f"{expected_fname} not found. This test is skipped.")
        skipped += 1
        continue

    checked += 1
    s = test+"..."
    print(f"{s.ljust(35)}", end='')

    test_path = os.path.join(tests_dir, test)
    fname = get_fname(test)

    llvm = subprocess.run([f"./{MATCAT}", test_path],
                          stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    if llvm.returncode != 0:
        if not is_positive_test:
            passed += 1
            print_g("PASSED. Failed at semantic checking.")
        else:
            print_r("FAILED")
        continue

    llvm_ir = llvm.stdout.decode()
    with open(f"{fname}.ll", "w") as f:
        f.write(llvm_ir)

    llvm_compile = subprocess.run([LLC, "-relocation-model=pic", f"{fname}.ll"],
                                  stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    if llvm_compile.returncode != 0:
        s = llvm_compile.stderr.decode()
        if not is_positive_test:
            passed += 1
            print_g("PASSED. Failed at llvm compile.")
        else:
            print_r("FAILED")

        clean_up(fname)
        continue

    # a *.s file should be generated at this point

    # Matrix Library is linked hardcoded here:
```

```
    clang_compile = subprocess.run([CC, "-o", f"{fname}.exe", f"{fname}.s",
"matrixLibrary.o", "-lm"],
                                   stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    if clang_compile.returncode != 0:
        s = clang_compile.stderr.decode()
        if not is_positive_test:
            passed += 1
            print_r("PASSED, clang compile failed, maybe sth is wrong.")
        else:
            print_r("FAILED, clang compile failed.")

        clean_up(fname)
        continue

    # a *.exe file shoulbe be generated at this point

    if not is_positive_test:
        print_r("FAILED, it should not compile.")
        clean_up(fname)
        continue

    # compare their results
    run_exe = subprocess.run([f"./{fname}.exe"],
                             stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    if clang_compile.returncode != 0:
        print_r("FAILED. Failed at running time. Maybe something is wrong")
        clean_up(fname)
        continue

    run_result = run_exe.stdout.decode().rstrip()
    test_op_path = os.path.join(tests_dir, f"{fname}{OUT}")
    expected_result = open(test_op_path).read().rstrip()
    if run_result != expected_result:
        print_r("FAILED. Output does not match.")

        # These are processed strings without trailing spaces
        print(f"\tExpected: {expected_result.encode()}")
        print(f"\tActual:   {run_result.encode()}")
        clean_up(fname)
        continue

    print_g("PASSED")
    passed += 1
    clean_up(fname)

print("\nTest summary:")
print_g(f"Passed: {passed}/{checked}")
print_r(f"Failed: {checked - passed}/{checked}")

if skipped:
    print_r(f"Skipped: {skipped}")
```

10.18 testall.sh

```
#!/bin/sh
# Authors: MicroC testall.sh modified by Davit

# Regression testing script for Matcat
# 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"

MATCAT="./matcat.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] [.mc 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/.mc//'`
    reffile=`echo $1 | sed 's/.mc$//'`
    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
${basename}.out" &&
    Run "$MATCAT" "$1" ">" "${basename}.ll" &&
    Run "$LLC" "-relocation-model=pic" "${basename}.ll" ">" "${basename}.s" &&
    Run "$CC" "-o" "${basename}.exe" "${basename}.s" "matrixLibrary.o" "-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/.mc//'`
    reffile=`echo $1 | sed 's/.mc$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

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

    generatedfiles=""

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

    # Report the status and clean up the generated files
```

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

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

shift `expr $OPTIND - 1`

LLIFail() {
  echo "Could not find the LLVM interpreter \"$LLI\"."
  echo "Check your LLVM installation and/or modify the LLI variable in testall.sh"
  exit 1
}

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


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

10.19 Integration Tests Files (Positive tests)

The following are the positive test cases of our integration test.

10.19.1 test-addFloats.mc

```
// Author: Davit
func main() int {
    double a;
    double b;
    double c;

    a = 5.1;
    b = 7.2;
    c = a + b;
    printd(c);

    return 0;
}
```

10.19.2 test-addFloats.out

```
12.3
```

10.19.3 test-addInts.mc

```
// Author: Davit
func main() int {
    int a;
    int b;
    int c;

    a = 5;
    b = 7;
    c = a + b;
    print(c);

    return 0;
}
```

10.19.4 test-addInts.out

```
12
```

10.19.5 test-addMatrix.mc

```
// Author: Davit
func main() int {
    matrix a;
    matrix b;
    matrix c;

    a = [[1,2,3],[3,4,5],[1,2,3]];
    b =[[3,2,1],[2,4,6],[1,2,3]];
    c = a + b;
    printm(c);
    return 0;
}
```

10.19.6 test-addMatrix.out

```
(
[4.00 4.00 4.00 ]
[5.00 8.00 11.00 ]
[2.00 4.00 6.00 ]
)
```

10.19.7 test-boolAssign.mc

```
// Author: James
func main() int
{

    bool b = false;

    printb(b);

    return 0;
}
```

10.19.8 test-boolAssign.out

```
0
```

10.19.9 test-comparison.mc

```
// Authors: Davit, James
func main() int
{

    bool b1;
    bool b2;
    bool r;
    int a;
    int b;
    bool geq;
    bool neq;
    bool lt;


    b1 = false;
    b2 = true;
    neq = b1 != b2;
    printb(neq); // 1
    a = 1;
    b = 0;
    geq = a >= b;
    lt = a < b;

    printb(geq); // 1
    printb(lt); // 0

    return 0;
}
```

10.19.10 test-comparison.out

```
1
1
0
```

10.19.11 test-demomatcat.mc

```
// Author: Davit
func main() int{
    matrix m;
    matrix c;
    matrix t;
    double d;
    matrix rhs;
    matrix lhs;
    double dp;
    matrix p;

    m = [[1,2,-1],[-2,0,1],[1,-1,0]];
    rhs = [[1,2,3,4]];
    lhs = [[2,5,7,5]];
    printStr("matrix m:");
    printm(m);
    printStr("");
    printStr("============================");

    //invertible
    isInv(m);

    c=inv(m);
    printStr("inverse of m is c:");
    printm(c);

    printStr("");
    printStr("============================");

    t=transpose(m);
    printStr("transpose of m is t:");
    printm(t);

    printStr("");
    printStr("============================");

    d=det(m);
    printStr("determinant of m is");
    printd(d);

    printStr("");
    printStr("============================");


    dp=rhs dot lhs;
    printStr("dot product of");
    printm(rhs);
    printStr("and");
    printm(lhs);
    printd(dp);

    printStr("");
    printStr("============================");

    p=m^5;
    printStr("m in power 5 is:");
    printm(p);

    printStr("");
    printStr("============================");
```

```
    rotate90(m);
    printStr("m rotated by 90 is:");
    printm(m);



    return 0;
});
    printm(m);



    return 0;
}
```

10.19.12 test-demomatcat.out

```
matrix m:
(
[1.00 2.00 -1.00 ]
[-2.00 0.00 1.00 ]
[1.00 -1.00 0.00 ]
)

============================
This matrix is invertible
inverse of m is c:
(
[1.00 1.00 2.00 ]
[1.00 1.00 1.00 ]
[2.00 3.00 4.00 ]
)

============================
transpose of m is t:
(
[1.00 -2.00 1.00 ]
[2.00 0.00 -1.00 ]
[-1.00 1.00 0.00 ]
)

============================
determinant of m is
1

============================
dot product of
(
[1.00 2.00 3.00 4.00 ]
)
and
(
[2.00 5.00 7.00 5.00 ]
)
53

============================
m in power 5 is:
(
```

```
[66.00 -52.00 1.00 ]
[32.00 45.00 -51.00 ]
[-41.00 11.00 15.00 ]
)

============================
m rotated by 90 is:
(
[-1.00 2.00 1.00 ]
[1.00 0.00 -2.00 ]
[0.00 -1.00 1.00 ]
)
```

10.19.13 test-detMatrix.mc

```
// Author: Davit
func main() int {
    matrix a;
    double d;
    a = [[1,0,1,1,2],[2,1,0,1,2],[3,1,2,1,2],[1,2,1,3,2],[1,4,5,2,2]];
    d = det(a);

    printd(d);

    return 0;
}
```

10.19.14 test-detMatrix.out

```
52
```

10.19.15 test-detMatrix1.mc

```
// Author: Davit
func main() int {
    matrix a;
    double d;
    a = [[1,2],[5,2]];
    d = det(a);

    printd(d);

    return 0;
}
```

10.19.16 test-detMatrix1.out

```
-8
```

10.19.17 test-dotProduct.mc

```
// Author: Davit
func main() int{
    matrix a;
    matrix b;
    double c;

    a = [[1,2,3]];
    b = [[2,3,4]];

    c = a dot b;
```

```
    printd(c);


    return 0;
}
```

10.19.18 test-dotProduct.out

```
20
```

10.19.19 test-dotProduct1.mc

```
// Author: Davit
func main() int{
    matrix a;
    matrix b;
    double c;

    a = [[1,2],[3,2]];
    b = [[2,3],[2,1]];

    c = a dot b;
    printd(c);


    return 0;
}
```

10.19.20 test-dotProduct1.out

```
16
```

10.19.21 test-for1.mc

```
// Author: James
func main() int
{
  int i;

  for (i = 0; i < 1 ; i = i + 1) {
      printStr("Loop Success");
  }

  return 0;
}
```

10.19.22 test-for1.out

```
Loop Success
```

10.19.23 test-for2.mc

```
// Authors: Davit, James
func main() int
{
  int i;
  for (i = 0; i < 1; i = i + 1) { // declare i inside loop
    printStr("Loop Success");

  }
```

```
  return 0;
}
```

10.19.24 test-for2.out

```
Loop Success
```

10.19.25 test-for3.mc

```
// Authors: Davit, James
func main() int
{
  int i;
  int j;
  for (i = 0; i < 1 ; i = i + 1) {
    for (j = 0; j <= i ; j = j + 1) { // declare i inside loop
      printStr("nested loop");

    }
  }

  return 0;
}
```

10.19.26 test-for3.out

```
nested loop
```

10.19.27 test-ifelse1.mc

```
// Authors: Davit, James
func main() int
{

  int f = 2;

  if(f > 0) {
      printStr("greater");
  } else {
      printStr("lesser");
  }

  return 0;
}
```

10.19.28 test-ifelse1.out

```
greater
```

10.19.29 test-ifelse2.mc

```
// Authors: Davit, James
func main() int
{

  bool b;
  b = true;
```

```
  if(b) {
      printStr("goodprint");
  }

  if(!b) {
      printStr("badprint");
  }


  return 0;
}
```

10.19.30 test-ifelse2.out

```
goodprint
```

10.19.31 test-ifelse3.mc

```
// Author: James
func main() int
{

  bool b = false;

  if(b) {
      printStr("badprint1");
  } else if(true) {
      printStr("goodprint");
  } else {
      printStr("badprint2");
  }

  return 0;
}
```

10.19.32 test-ifelse3.out

```
goodprint
```

10.19.33 test-inverseMatrix.mc

```
// Author: Davit
func main() int
{
    matrix a;
    matrix c;
    a = [[1,2],[4,5]];
    c = inv(a);

    printm(c);

    return 0;

}
```

10.19.34 test-inverseMatrix.out

```
(
[-1.67 0.67 ]
[1.33 -0.33 ]
```

```
)
```

10.19.35 test-inverseMatrix1.mc

```
// Author: Davit
func main() int
{
    matrix a;
    matrix c;
    a = [[2,5],[5,1]];
    c = inv(a);

    printm(c);

    return 0;

}
```

10.19.36 test-inverseMatrix1.out

```
(
[-0.04 0.22 ]
[0.22 -0.09 ]
)
```

10.19.37 test-inverseMatrix2.mc

```
// Author: Davit
func main() int
{
    matrix a;
    matrix c;
    a = [[2,5,1],[5,1,6],[3,1,2]];
    c = inv(a);

    printm(c);

    return 0;

}
```

10.19.38 test-inverseMatrix2.out

```
(
[-0.12 -0.26 0.85 ]
[0.24 0.03 -0.21 ]
[0.06 0.38 -0.68 ]
)
```

10.19.39 test-isInvertible.mc

```
// Author: Davit
func main() int {
    matrix a;
    a = [[3,3],[5,5]];
    isInv(a);

    return 0;
}
```

10.19.40 test-isInvertible.out

```
This matrix is not invertible
```

10.19.41 test-isInvertible1.mc

```
// Author: Davit
func main() int {
    matrix a;
    a = [[5,6],[8,8]];
    isInv(a);

    return 0;
}
```

10.19.42 test-isInvertible1.out

```
This matrix is invertible
```

10.19.43 test-manyPrints.mc

```
// Author: James
func main() int
{
    printStr("hello world");

    printStr("with more words");
    return 0;

}
```

10.19.44 test-manyPrints.out

```
hello world
with more words
```

10.19.45 test-matrix-function1.mc

```
// Author: Andreas
func m() matrix {
        return [[3, 4]];
}

func main() int {
    matrix m;
    m = m();
    printm(m);

    return 0;
}
```

10.19.46 test-matrix-function1.out

```
(
[3.00 4.00 ]
)
```

10.19.47 test-matrixAccess1D.mc

```
// Author: Davit
func main() int{
    matrix a;
    a = [[1,2,3],[4,5,6]];
    printm(a[1,:]);

    return 0;
}
```

10.19.48 test-matrixAccess1D.out

```
(
[4.00 5.00 6.00 ]
)
```

10.19.49 test-matrixAccess2D.mc

```
// Author: Davit
func main() int {
    matrix c;
    c = [[1,2],[3,4]];
    printm(c[0][0]);

    return 0;
}
```

10.19.50 test-matrixAccess2D.out

```
(
[1.00 ]
)
```

10.19.51 test-matrixAccessCol.mc

```
// Author: Davit
func main() int{
    matrix a;
    a = [[1,2,3],[4,5,6],[10,20,34],[3,7,6]];
    printm(a[:,1]);

    return 0;
}
```

10.19.52 test-matrixAccessCol.out

```
(
[2.00 5.00 20.00 7.00 ]
)
```

10.19.53 test-matrixAccessDiagonal.mc

```
// Author: Davit
func main() int {
    matrix a;
    matrix b;

    a = [[1,2,3],[4,5,6],[7,8,9]];
    b = a[:,:];
```

```
    printm(b);


    return 0;
}
```

10.19.54 test-matrixAccessDiagonal.out

```
(
[1.00 5.00 9.00 ]
)
```

10.19.55 test-matrixAdd1.mc

```
// Author: Davit
func main() int {
    matrix c;
    c = [[1,2],[3,4]] + [[3,2],[5,4]];
    printm(c);

    return 0;
}
```

10.19.56 test-matrixAdd1.out

```
(
[4.00 4.00 ]
[8.00 8.00 ]
)
```

10.19.57 test-matrixMult.mc

```
// Author: Davit
func main() int {
    matrix a;
    matrix b;
    matrix c;

    a = [[1,2,3],[4,5,6],[7,8,9]];
    b = [[10,11,12],[13,14,15],[16,17,18]];

    c = a * b;
    printm(c);

    return 0;
}
```

10.19.58 test-matrixMult.out

```
(
[84.00 90.00 96.00 ]
[201.00 216.00 231.00 ]
[318.00 342.00 366.00 ]
)
```

10.19.59 test-matrixMult1.mc

```
// Author: Davit
func main() int {
```

```
    matrix a;
    matrix c;

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


    c = a * a;
    printm(c);

    return 0;
}
```

10.19.60 test-matrixMult1.out

```
(
[9.00 12.00 ]
[24.00 33.00 ]
)
```

10.19.61 test-matrixPower.mc

```
// Author: Davit
func main() int {
    matrix a;
    matrix b;
    a = [[1,2,3],[4,5,6],[7,8,9]];
    b = a^5;

    printm(b);

    return 0;

}
```

10.19.62 test-matrixPower.out

```
(
[121824.00 149688.00 177552.00 ]
[275886.00 338985.00 402084.00 ]
[429948.00 528282.00 626616.00 ]
)
```

10.19.63 test-matrixPower1.mc

```
// Author: Davit
func main() int {
    matrix a;
    matrix b;
    a = [[1,2,3],[4,5,6],[7,8,9]];
    b = a^0;

    printm(b);

    return 0;

}
```

10.19.64 test-matrixPower1.out

```
(
[1.00 0.00 0.00 ]
```

```
[0.00 1.00 0.00 ]
[0.00 0.00 1.00 ]
)
```

10.19.65 test-matrixPower2.mc

```
// Author: Davit
func main() int {
    matrix a;
    matrix b;
    a = [[1,2,2],[4,5,6],[7,0,9]];
    b = a^-3;

    printm(b);

    return 0;

}
```

10.19.66 test-matrixPower2.out

```
(
[-35.18 13.25 -0.99 ]
[-3.40 1.29 -0.11 ]
[26.45 -9.97 0.75 ]
)
```

10.19.67 test-matrixSub.mc

```
// Author: Davit
func main() int {
    matrix a;
    matrix b;
    matrix c;

    a = [[1,2,3],[3,4,5],[1,2,3]];
    b =[[3,2,1],[2,4,6],[1,2,3]];
    c = a - b;
    printm(c);
    return 0;
}
```

10.19.68 test-matrixSub.out

```
(
[-2.00 0.00 2.00 ]
[1.00 0.00 -1.00 ]
[0.00 0.00 0.00 ]
)
```

10.19.69 test-matrixSymmetry.mc

```
// Author: Davit
func main() int {
    int test;
    matrix a = [[1,2,3],[2,4,5],[3,5,6]];
    matrix b;

    test = check_symmetry(a);
    if(test == 1){
        printStr("The matrix is symmetric");
```

```
        printStr("Here is the original matrix:");
        printm(a);
        printStr("");
        printStr("Here is the transposed matrix:");
        b = transpose(a);
        printm(b);

    }else{
        printStr("The matrix is not symmetric");
    }
}
```

10.19.70 test-matrixSymmetry.out

```
The matrix is symmetric
Here is the original matrix:
(
[1.00 2.00 3.00 ]
[2.00 4.00 5.00 ]
[3.00 5.00 6.00 ]
)

Here is the transposed matrix:
(
[1.00 2.00 3.00 ]
[2.00 4.00 5.00 ]
[3.00 5.00 6.00 ]
)
```

10.19.71 test-multipleAssignment.mc

```
// Author: James
func main() int {

    int a;
    int b;
    int c;
    a = b = c = 5;
    if(a == b && a == c) {
        printStr("equal");
    }
    else {
        printStr("oh no");
    }


    return 0;
}
```

10.19.72 test-multipleAssignment.out

```
equal
```

10.19.73 test-print.mc

```
// Author: Davit
func main() int
{
    printStr("hello world");
    return 0;
```

```
}
```

10.19.74 test-print.out

```
hello world
```

10.19.75 test-printMatrix.mc

```
// Author: Davit
func main() int {
    matrix m;
    m = [[1,2,3],[4,5,6],[10,20,30],[100,255,560]];
    printm(m);

    return 0;
}
```

10.19.76 test-printMatrix.out

```
(
[1.00 2.00 3.00 ]
[4.00 5.00 6.00 ]
[10.00 20.00 30.00 ]
[100.00 255.00 560.00 ]
)
```

10.19.77 test-printb.mc

```
// Author: Davit
func main() int {

    printb(true);
    return 0;

}
```

10.19.78 test-printb.out

```
1
```

10.19.79 test-printstring.mc

```
// Author: James
func main() int
{
    printStr("hello world");
    return 0;
}
```

10.19.80 test-printstring.out

```
hello world
```

10.19.81 test-printvoid.mc

```
// Author: Davit
func printVoid() void {
    printStr("Hello world");
}
```

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

10.19.82 test-printvoid.out

```
Hello world
```

10.19.83 test-recursion.mc

```
// Author: James
func foo(int n) int {
    if (n == 1) {
        return 0;
    }
    return bar(n - 1);
}

func bar(int n) int {
    if (n == 1) {
        return 0;
    }
    return foo(n - 1);
}

func main() int {
    print(foo(5));
    return 0;
}
```

10.19.84 test-recursion.out

```
0
```

10.19.85 test-redeclareFuncParam.mc

```
// Author: Davit
func foo(int x) int {
    int x;
    x = 2;
    return x;
}


func main() int {

    print(foo(5));

    return 0;
}
```

10.19.86 test-redeclareFuncParam.out

```
2
```

10.19.87 test-rotate90.mc

```
// Author: Davit
func main() int {
    matrix a = [[1,2,3],[4,5,6],[7,8,9]];
    rotate90(a);
    printm(a);
    return 0;
}
```

10.19.88 test-rotate90.out

```
(
[7.00 4.00 1.00 ]
[8.00 5.00 2.00 ]
[9.00 6.00 3.00 ]
)
```

10.19.89 test-rowAccessRetMatrix.mc

```
// Author: James
func main() int {
    matrix m;
    m = [[1,2,3]];

    int i = 0;
    while(i < 3) {
        i = i+1;
        m = m[0,:];
        printm(m);
    }


    return 0;
}
```

10.19.90 test-rowAccessRetMatrix.out

```
(
[1.00 2.00 3.00 ]
)
(
[1.00 2.00 3.00 ]
)
(
[1.00 2.00 3.00 ]
)
```

10.19.91 test-scalarDivDouble.mc

```
// Author: Davit
func main() int {
    matrix a;
    matrix b;

    a = [[1,2,3],[4,5,6],[2,10,2]];
    b = a / 2.5;

    printm(b);

    return 0;
```

```
}
```

10.19.92 test-scalarDivDouble.out

```
(
[0.40 0.80 1.20 ]
[1.60 2.00 2.40 ]
[0.80 4.00 0.80 ]
)
```

10.19.93 test-scalarDivMatrix.mc

```
// Author: Davit
func main() int {
    matrix a;
    matrix b;

    a = [[1,2,3],[4,5,6],[2,10,2]];
    b = a / 2;

    printm(b);

    return 0;

}
```

10.19.94 test-scalarDivMatrix.out

```
(
[0.50 1.00 1.50 ]
[2.00 2.50 3.00 ]
[1.00 5.00 1.00 ]
)
```

10.19.95 test-scaleMatrix.mc

```
// Author: Davit
func main() int {
    matrix a;
    matrix b;

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

    printm(b);

    return 0;
}
```

10.19.96 test-scaleMatrix.out

```
(
[3.00 6.00 9.00 ]
[12.00 15.00 18.00 ]
)
```

10.19.97 test-scaleMatrix1.mc

```
// Author: Davit
```

```
func main() int {
    matrix a;
    matrix b;

    a = [[1,2,3],[4,5,6],[2,10,2]];
    b = 1 * a;

    printm(b);

    return 0;

}
```

10.19.98 test-scaleMatrix1.out

```
(
[1.00 2.00 3.00 ]
[4.00 5.00 6.00 ]
[2.00 10.00 2.00 ]
)
```

10.19.99 test-scaleMatrixDouble.mc

```
// Author: Davit
func main() int {
    matrix a;
    matrix b;

    a = [[1,2,3],[4,5,6],[2,10,2]];
    b = 2.5 * a;

    printm(b);

    return 0;

}
```

10.19.100 test-scaleMatrixDouble.out

```
(
[2.50 5.00 7.50 ]
[10.00 12.50 15.00 ]
[5.00 25.00 5.00 ]
)
```

10.19.101 test-while1.mc

```
// Author: James
func main() int
{

    bool b = false;

    while(b) {}

    printStr("pass");

    return 0;
}
```

10.19.102 test-while1.out

```
pass
```

10.19.103 test-while2.mc

```
// Author: James
func main() int
{

    while(2 != (2 * 1)) {}

    printStr("pass");

    return 0;
}
```

10.19.104 test-while2.out

```
pass
```

10.20 Integration Tests Files (Negative tests)

The following are the negative test cases of our integration test.

10.20.1 fail-assign1.err

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

10.20.2 fail-assign1.mc

```
// Author: James
func main() int
{
  int i;
  bool b;

  i = 42;
  i = 10;
  b = true;
  b = false;
  i = false; /* Fail: assigning a bool to an integer */
  return 0;
}
```

10.20.3 fail-assign2.err

```
Fatal error: exception Failure("illegal assignment bool = int in b = 48")
```

10.20.4 fail-assign2.mc

```
// Author: James
func main() int {
  int i;
  bool b;

  b = 48; /* Fail: assigning an integer to a bool */
  return 0;
}
```

10.20.5 fail-assign3.err

```
Fatal error: exception Failure("illegal assignment int = void in i = myvoid()")
```

10.20.6 fail-assign3.mc

```
// Author: James
func myvoid() void
{
  return;
}

func main() int
{
  int i;

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

10.20.7 fail-binop-floats-ints.err

```
Fatal error: exception Failure("illegal binary operator int + double in i + a")
```

10.20.8 fail-binop-floats-ints.mc

```
// Author: James
func main() int {

    int i;
    double a;
    double b;
    double c;

    i = 5;
    a = 6.0;
    c = i + a; // still debatable whether we support this

    printd(c);


/*
    int a;
    a = 5;

    double b;
    b = 6.1;

    double r;
    r = a + b;

    if (r == 11) {
        printStr("oh no");
    } else {
        printStr("okay");
    }
*/
    return 0;
}
```

10.20.9 fail-call-nonfunc.err

```
fail-call-nonfunc.err
```

10.20.10 fail-call-nonfunc.mc

```
// Author: James
func main() int {

    int foo;
    foo = 5;
    foo();

    return 0;
}
```

10.20.11 fail-emptyMatrixDeclare.err

```
Fatal error: exception Failure("hd")
```

10.20.12 fail-emptyMatrixDeclare.mc

```
// Author: James
func main() int {
    matrix m;
    m = [[], [], []];
    printm(m);

    return 0;
}
```

10.20.13 fail-for1.err

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

10.20.14 fail-for1.mc

```
// Author: James
func main() int
{
  int i;
  for ( ; true ; ) {} /* OK: Forever */

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

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

  return 0;
}
```

10.20.15 fail-for2.err

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

10.20.16 fail-for2.mc

```
// Author: James
```

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

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

  return 0;
}
```

10.20.17 fail-for3.err

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

10.20.18 fail-for3.mc

```
// Author: James
func main() int
{
  int i;

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

  return 0;
}
```

10.20.19 fail-for4.err

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

10.20.20 fail-for4.mc

```
// Author: James
func main() int
{
  int i;

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

  return 0;
}
```

10.20.21 fail-for5.err

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

10.20.22 fail-for5.mc

```
// Author: James
func main() int
{
  int i;

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

  return 0;
}
```

10.20.23 fail-funcInFunc.err

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

10.20.24 fail-funcInFunc.mc

```
// Author: James
func main() int {



    func foo() int {
        return 0;
    }



    return 0;
}
```

10.20.25 fail-funcInLoop.err

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

10.20.26 fail-funcInLoop.mc

```
// Author: James
func main() int {

    while(true) {

        func foo() int {
            return 0;
        }

    }

    return 0;
}
```

10.20.27 fail-helloworld1.err

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

10.20.28 fail-helloworld1.mc

```
// Authors: Davit, James
func main() int
{
    printStr("hell) // malformed
    return 0;
}
```

10.20.29 fail-ifWithoutExpr.err

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

10.20.30 fail-ifWithoutExpr.mc

```
// Author: James
func main() int {

    if () {
        printStr("no");
    }

    return 0;
}
```

10.20.31 fail-ifelse1.err

```
Fatal error: exception Failure("expected Boolean expression in 5")
```

10.20.32 fail-ifelse1.mc

```
// Author: James
func main() int
{

  bool b = true;

  if(5) { // non bool
      printStr("goodprint");
  }

  if(!b) {
      printStr("badprint");
  }

  return 0;
}
```

10.20.33 fail-ifelse2.err

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

10.20.34 fail-ifelse2.mc

```
// Author: James
func main() int
{

  bool b;
  b = true;

  if(b) {
      printStr("hm");
  } else {
      printStr("end?");
  } else if(false) { // elif after else
      printStr("blah");
  }

  return 0;
}
```

10.20.35 fail-ifelse3.err

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

10.20.36 fail-ifelse3.mc

```
// Author: James
func main() int
{

  bool b;
  b = true;

  if(b) {
      if(!b) {
          if(true) {
              else { // nested but no corresponding if
                  printStr("very nested")
              }
          }
      }
  }

  return 0;
}
```

10.20.37 fail-illegalVarName.err

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

10.20.38 fail-illegalVarName.mc

```
// Author: James
func main() int {

    int $a;
    int 1a;
    1a = 2;

    return 0;
}
```

10.20.39 fail-multipleTypes.err

```
Fatal error: exception Failure("duplicate local a")
```

10.20.40 fail-multipleTypes.mc

```
// Author: James
func main() int {

    int a;
    double a;

    a = 5;
    a = 7.2;

    return 0;

}
```

10.20.41 fail-nestedMatrix.err

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

10.20.42 fail-nestedMatrix.mc

```
// Author: James
func main() int {
    matrix m;
    m = [[[1,2,3], [4, 5, 6]], [7, 8, 9]];
    printm(m);

    return 0;
}
```

10.20.43 fail-returnTypeMismatch.err

```
Fatal error: exception Failure("return gives double expected int in 5.0")
```

10.20.44 fail-returnTypeMismatch.mc

```
// Author: James
func main() int {

    int a;
    a = 1;

    return 5.0;
}
```

10.20.45 fail-topLevelReturn.err

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

10.20.46 fail-topLevelReturn.mc

```
// Author: James
return;

func main() int {

    return 0;
}
```

10.20.47 fail-while1.err

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

10.20.48 fail-while1.mc

```
// Author: James
func main() int
{

    int i;
    i = 0;
    while(i) { } // non bool

    return 0;
```

```
}
```

10.21 Future Work Cases

These are bugs which we were not able to sort out before the deadline, but we are nonetheless aware of.

10.21.1 fail-badMatrixIndex.err

```
pass
```

10.21.2 fail-badMatrixIndex.mc

```
// Author: James, Andreas (Renaming only)
func main() int {
    matrix a;
    a = [[1,2,3]];
    printm(a[1,:]); // runtime error, will not work with diff

    return 0;
}
```

10.21.3 fail-noReturn.err

```
pass
```

10.21.4 fail-noReturn.mc

```
// Author: James, Andreas (Renaming only)
func main() int {
    matrix a;
    a = [[1,2,3]];
    printm(a[1,:]); // runtime error, will not work with diff

    return 0;
}
```

10.21.5 test-assignMatrixTypes.mc

```
// Author: James, Andreas (Renaming only)
func main() int {
    matrix a;
    a = [[1,2,3]];
    printm(a[1,:]); // runtime error, will not work with diff

    return 0;
}
```

10.21.6 test-assignMatrixTypes.out

```
// Author: James, Andreas (Renaming only)
func main() int {
    matrix a;
    a = [[1,2,3]];
    printm(a[1,:]); // runtime error, will not work with diff

    return 0;
}
```

10.21.7 test-dotproduct-different-dim.mc

```
// Author: James, Andreas (Renaming only)
func main() int {
    matrix a;
    a = [[1,2,3]];
    printm(a[1,:]); // runtime error, will not work with diff

    return 0;
}
```

10.21.8 test-dotproduct-different-dim.out

```
// Author: James, Andreas (Renaming only)
func main() int {
    matrix a;
    a = [[1,2,3]];
    printm(a[1,:]); // runtime error, will not work with diff

    return 0;
}
```

10.21.9 test-transposeMatrix.mc

```
// Author: James, Andreas (Renaming only)
func main() int {
    matrix a;
    a = [[1,2,3]];
    printm(a[1,:]); // runtime error, will not work with diff

    return 0;
}
```

10.21.10 test-transposeMatrix.out

```
// Author: James, Andreas (Renaming only)
func main() int {
    matrix a;
    a = [[1,2,3]];
    printm(a[1,:]); // runtime error, will not work with diff

    return 0;
}
```