MathLight: Lightweight MATLAB implementation
Final Report

Boya Song (bs3065),
Chunli Fu (cf2710),
Mingye Chen (mc4414),
Yuli Han (yh2986)

Tables of Content

Introduction **3**

Tutorial **3**

2.1 Environment setup 3

2.2 Language Tutorial 4

Reference Manual **6**

3.1 Data Types 6

3.2 Lexical Conventions 7

3.3. Syntax Notation 9

3.4 Declaration 10

3.5 Statements 11

3.6 Control Flows 12

3.7 Built-in Functions 13

Project Plan **14**

4.1 Planning Process 14

4.2 Specification Process 15

4.3 Development Process 15

4.4 Testing Process 15

4.5 Style Guide 16

4.6 Roles and Responsibilities 16

4.7 Software Development Environments 16

4.8 Timeline 17

4.9 Project Log 17

Architectural Design **35**

5.1 Architecture Overview 35

5.2 Scanner (Mingye, Yuli) 36

5.3 Parser (Mingye) 36

5.4 Semantic Checker (Chunli) 36

5.5 Code Generator (Boya, Yuli) 36

Test Plan **37**

6.1 Representative programs 37

6.2 Test Suites 60

6.3 Test Coverage 60

6.4 Test Automation 60

6.5 Roles 60

Language Evolution **60**

Appendix **62**

1. Introduction

MathLight is a lightweight implementation of MATLAB. As a programming language, it allows basic matrix manipulations and common statistics computations. Our goal is to make it as an easy, fast and flexible language.

In order to make it powerful and swift, we add the matrix data type since it's widely used in the areas of scientific computations. Also, many arithmetic operators and relational operators are designed for matrix manipulations such as matrix multiplication, transpose, dot multiplication and so on. We implemented rich built-in functions to make it user-friendly: statistics computations such as mean, sum and more are supported, matrix computations like determinant, norm, trace, eigenvalues are prepared for developers to use.

The syntax of mathlight is like C so it is very easy to learn and use. We use static scoping and static type.

2. Tutorial

2.1 Environment setup

To use our language, firstly get the whole project from the zip file we submitted. It needs the OCaml llvm library, which is most easily installed through opam.

Then install LLVM and its development libraries, the m4 macro preprocessor and opam, and use opam to install llvm.

The version of the OCaml llvm library must match the version of the LLVM system installed on your system. At the time of the development, we use ocaml 4.07 and llvm 7.0.0.

You need to use '**make**' to compile the project. It will also run all the test cases automatically. To compile your own program, use the command '**./mathlight.native [program_name].txt > [program_name].ll**' to produce a .ll file, then use '**llc -relocation-model=pic [program_name].ll > [program_name].s**' and **'cc -o [program_name].exe [program_name].s matrices.o'** to generate the executable '**[program_name].exe**'. To run your program, run the command '**./[program_name].exe**'.

2.2 Language Tutorial

2.2.1 Variable Declaration

Our language support six kinds of basic data types, **int , double, string, boolean, void and matrix** (void is only used in function returns and cannot be declared). You can see the details in language reference manual. To declare a variable, use syntax like:

```
int a;
matrix b<2,2>;
matrix c<3>;
```

You need to specify the size of matrix when declare a matrix. For 1-D matrix, you only need to provide one dimension.

You can also do the declaration and assignment at the same time, like:

```
matrix b<2,2> = [1,2;3,4];
matrix c<3> = fill(1,3,1.0);
```

This statement will create a matrix with two rows and two columns and assign the value for each element. 'fill' is a built-in function to initialize the matrix with given size and given default value.

2.2.2 Function Declaration

You can write function in our language. You need to use the below syntax to define a function.

func return_type function_name(type1 arg1, type2, arg2) {

statements.
}

The following code is an example to declare a function which add two matrices and return the result matrix. You **do not** need to specify the size for matrices when you use matrices as formal arguments and return type.

```
func matrix mat_add(matrix a, matrix b) {
	return a+b;
}
```

2.2.3 Control Flow

The control flow of our language is very similar to C. We support if conditional statement and for loop and while loop statements. The following code shows an example to use if to do control flow.

```
if (1>2) print("first");
else if (3>4) print("second");
else print("final");
```

The following code is an example to use for loop in our language.

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

The following code is an example to use while loop in our language.

```
func int main() {
  int i;
  i = 0;
  while (i < 5) {
     print(i);
     i = i + 1;
  }
  return 0;
}
```

2.2.4 Built-in functions and arithmetic operators

Our language has very powerful matrix-related built-in functions as well as the arithmetic operations. You can see all the built-in functions and they introduction in language reference manual. As an instance, if you want to compute the inverse matrix for a square matrix, you can use the below code.

```
matrix a<2,2> = [1,2;3,4];
matrix b<2,2> = inv(a);
```

Our arithmetic operations support different type. You can add two integers, two doubles, two matrices. You can even add an integer to a double. The integer will be casted to double and return a double value. You can add an integer or a double value to a matrix using the below code.

```
matrix a<2,2> = [1,2;3,4];
int b = 1;
a = a + b;
```

3. Reference Manual

3.1 Data Types

There are six kinds of basic data types in our language. The declaration of them are similar to statically-typed language like C.

Type names	Description
int	32-bit signed integer
double	64- bit double precision float-point number
boolean	1-bit logical value
string	string data
matrix	one or two dimensional matrix data with double type
void	no type

3.2 Lexical Conventions

There are six kinds of tokens: identifiers, keywords, constants, strings, expression operators, and other separators. In general, blanks, tabs, newlines, and comments as described below are ignored except as they serve to separate tokens. At least one of these characters is required to separate otherwise adjacent identifiers, constants, and certain operator-pairs.

3.2.1 Identifiers

In our language, identifiers uniquely defines an object. Identifiers must start with lowercase letter, followed by any combinations of numbers, letters and underscores.

3.2.2 Keywords

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

int double bool string matrix void True False
if else for while continue break func return PI

3.2.3 Constants

Numbers, strings are constants. Besides, predefined values used for quick access in mathematic computation, such as PI, are also constants.

3.2.4 Strings

A string is a sequence of characters surrounded by double quotes ‘‘ " ’’.

3.2.5 Expression Operators

3.2.5.1 Arithmetic operators

Operators	Description
+	Addition(int, double, matrix)

-	Subtraction(int, double, matrix)
*	Multiplication(int, double, matrix)
/	Division(int, double)
^	Power(int, double)
||	Absolute value(int, double)
.*	Element-wise multiplication for matrix
./	Element-wise division for matrix
‘	Transpose for matrix

For add, minus and multiplication, we support operations between different types. Users can apply the plus operation between int and int, int and double, double and matrix, int and matrix. When they add/subtract/multiply an integer to a double, the integer will be casted to a double value and return a double. When they add/subtract/multiply a double to matrix, the operator will be applied to every element in the matrix with the double and return a new matrix.

The multiplication and division of the matrix are defined as that in the linear algebra. The dot is a special operator for matrix in our language. The dot means the operator is applied to each elements in the matrix.

3.2.5.2 Relational operators

Operators	**Description**
>	Greater than (int, double)
<	Small than(int, double)
<=	Smaller than or equal to (int, double)
>=	Greater than or equal to (int, double)
!=	Not equal to (int, double, matrix)
==	Equal to (int, double)

Two matrix are identical if they have the same size and the elements at the corresponding positions are the same.

3.2.5.3 Logical operators

Operators	Description
!	Not(boolean)
&&	And(boolean)
||	Or(boolean)

3.2.5.4 Other operators

Operators	Description
[]	Concatenate matrices horizontally, eg: [matrix a, matrix b] Concatenate matrices vertically, eg: [matrix a**;** matrix b]

3.2.6 Comments

In MathLight, the comments are similar to that in C language. We use /* to start a comment and */ to end it. Anything between them is ignored and comments cannot be nested. Besides, we also can use // to start a comment in a single line.

3.3. Syntax Notation

3.3.1 Primary Expression

Primary expressions are most basic expressions, which make up complex expressions. It includes constants, identifiers and expressions in parentheses.

3.3.2 Complex Expression

Expression	Explanation or Example
expression[expression]	such as a[1+2, 2*2] (a is an initialized 2d matrix)
expression(argument list)	function calls

identifier = expression	assignment

3.4 Declaration

3.4.1 Data Type Declaration and Initialization

Declarations and initializations for basic data types are similar to that in C as follow.

decl-specifiers declarator-list

The declarators in the declarator-list contain the identifiers being declared. The decl-specifiers consist of at most one type-specifier

Declaration	**Initialization**
int a;	int a = 0;
double a;	double a = 1.0;
boolean a;	boolean a = false;
string a;	string a = “hello world!”;

matrix a;	matrix a<2,4> = [1, 2, 3, 4; 1, 2, 3, 4];
	matrix a<2,4> = fill(2, 4, 1.0); (default value: 1.0)
	matrix b<3> = [1,2,3];

Specially, for matrix, the dimension is set to be 2 and the data type of elements in the matrix can only be double. Users need to specify the size of matrix in declaration. We also support vectors, which is an 1-D matrix.

We use the square brackets to initialize a matrix with values, eg, matrix a<2,4> = [1, 2, 3, 4; 1, 2, 3, 4];. The different rows of a matrix is separated by semicolon. While the different entries in one row is separated by comma. We can use syntax like a[1, 2] to get the value which locate at the intersection of the second row and the third column.

Or we can define a matrix with size and default value, such as matrix a<2,4> = fill(2, 4, 2.0);. We use parentheses to define a matrix with size and initialize it with the given default value.

3.4.2 Function Declaration

The user-defined function could be declared as below:

```
func T name(T arg, ...) {
        statements
}
```

It specifies the return type, arguments name and type and function name. For example:

```
func int add_num(int a, int b)  {
        return a + b;
}

func matrix mat_add(matrix a, matrix b)  {
        return a + b;
}
```

If nothing is returned, the return type is void.

3.5 Statements

In MathLight, every statement is followed by a semicolon, like :

statement;

3.5.1 Expression statements

Each expression statement includes one expression, which is usually an assignment or a function call.

3.5.2 Compound statements

A compound statement is a sequence of statements enclosed by braces as follow:

```
{
        statement1;
        statement2;
}
```

If a variable is declared in a compound statement, then the scope of this variable is limited into this statement.

3.6 Control Flows

3.6.1 Condition statements

The definition of a if statement is like below:

```
if  (condition(statement)) {
        Statements
}
else if (condition(statement)) {
        Statements
}
else {
        Statements
}
```

Statement following *if/ else if* (condition) must be boolean statement, or can be evaluated to boolean.

3.6.2 Loop statements

The definition for a for loop is like below:

```
for ( initializer(statement); condition(statement); iterator(statements)) {
        Statements
}
```

Here, initializer sets the initial state, condition is a boolean value controls whether continuing executing statements, and iterator is executed after each iteration. For example:

```
for (a = 1 ; a < 3; a = a + 1) {
        print(a)
}
```

The definition for a while loop is like below:

```
while (condition(statement)) {
        Statements
}
```

Here, condition is a boolean expression, which controls whether continuing executing the statements. For example:

```
while (a > 0) {
        a = a - 1
}
```

3.7 Built-in Functions

Function names	Description
fill(int r, int c, double default)	Initialize 2D matrix with size and default value
print(int num) print(double num) print(string s) print(matrix m)	Printing value on the screen.
sqrt(double num)	Compute the square root for a double value.
log(double num)	Compute the logarithm value of baes 2 for a double.
mean_row(matrix a) mean_col(matrix a)	Compute the mean value of a row or column in a matrix. For a matrix<n,m>, the mean_row(m) will return a matrix ret<n,1> and ret[n, 0] stores the mean value for the nth row.
sum_row(matrix a)	Compute the sum of a row or column in a matrix.

sum_col(matrix a)	For a matrix<n,m>, the sum_row(m) will return a matrix ret<n,1> and ret[n, 0] stores the sum value for the nth row.
inv(matrix m)	Get the Inverse matrix of a given a matrix
det(matrix m)	Compute the determinant of a matrix
tr(matrix m)	Compute the trace of a matrix
max_eigvalue(matrix m)	Compute the maximal eigenvalues of a real symmetric square matrix.
sizeof_row(matrix m) sizeof_col(matrix m)	Get the number of rows and columns in a matrix.
norm1(matrix m) norm2(matrix m)	The norm1 function returns the absolute norm for a vector or a matrix. The norm2 function returns the Euclidean norm for a matrix or a norm.

4. Project Plan

4.1 Planning Process

Our team is built right after the second class. All of the teammates are second-year master students and we are very familiar with each other so it took very short time to build this team. We created a Wechat group for better communication. We also created a Google Drive folder to store all the useful resource for our project. We added our project proposal, language reference manual and presentation slides to this folder after we finished them.

We decided to implement a matrix manipulation language in the first group meeting. All of our teammates are engineering students and used matrices frequently. We all agreed that designing a matrix manipulation language is a good idea. And we also think support some statistics feature and image processing would be useful. We wrote the project proposal together as we need to decide the basic syntax and grammar of our language.

4.2 Specification Process

As mentioned above, our project proposal included not only matrices manipulation but also some other functions, like the image processing, file I-O and some statistics function. But pointed out by TA, we realized that it's better for us to concentrate on one feature and implemented it well. So we decided to focus on matrices then.

4.3 Development Process

We began to work on the scanner and parser at the same time when we worked on the language reference manual. Then we started working on the Hello World program after that. We only did semantic checking and implemented the code generator related to Hello World program at first. When we ran the hello world program successfully, all of us felt a sense of accomplishment.

Then we continued working on the project after the Thanksgiving break and concentrated on it after the course final. All of the team members worked together everyday. We kept updating the language reference manual based on our implementation details. For instance, we used to have variable declaration and assignment separately in our language, which made the program too long. We modified the structure to make declaration and assignment together possible. After we finished the basic development of our project, we added many built-in functions to enrich the matrix manipulation function.

4.4 Testing Process

We used the integration test to test our code. Everytime we added a new feature in code generation file, we wrote at least one test cases for it. To test our semantic checker, we also wrote many failed cases to make sure the semant check works well.

We wrote a test script, i.e., testall.sh, to automatically run our test suites. The "make" instruction will build our compiler and run the test suites automatically. If the compiler is already built, we can run 'make test' or './testall.sh' to run all the test cases. By this way, we can figure out the cause for failures easily on time.

4.5 Style Guide

We have a style convention for our team, which is shown below.

1. Correct indentation for different blocks. Using tab size 2 for Ocaml code.
2. Underline method to name built-in function names.
3. Variable naming starts with lower case english letters.
4. Test files ended in .txt
5. Using descriptive name to naming functions.
6. Writing functions for any piece of code which has been used for more than three times

4.6 Roles and Responsibilities

Boya Song is the manager of our team and a tester. She was responsible for the integration of the whole project. She implemented the basic structure of codegen file. She also implemented the matrix inner structure, function, and some built-in functions. She also wrote some test cases.

Chunli Fu is the system architect as well as a tester of our project. She wrote all the semantic checking code and the corresponding test cases.

Mingye Chen is language guru and a tester. He designed the MathLight syntax and grammar. He also wrote the parser and some features of scanner for our language. He also wrote some test cases.

Yuli Han is a system architect and a tester. She implemented all the operators and some built-in functions. She also wrote the test cases to test these features. She also wrote some features of scanner.

4.7 Software Development Environments

Operating Systems: Mac OS Systems
Languages: Opam: 2.0.1, OCaml 4.0.7, LLVM 7.0.0

Text Editor: Sublime Text/VSCode
Version Control: Git
Collaboration: GitHub
Documentation: Google Doc

4.8 Timeline

Sep 10th - Sep 19th	Assembled team members and wrote the project proposal
Sep 20th - Oct 1st	Finished the language syntax and grammar design. Wrote the language reference manual.
Oct 1st - Oct 15th	Implemented scanner and parser. Fixed all the conflicts issues.
Oct 16th - Nov 14th	Implemented the basic semantic check and code generation code related to Hello World program.
Nov 15th - Dec 17th	Enriched the code generation, semant check. Modified the scanner, parser and the language reference manual based on the progress.
Dec 17th - Dec 18th	Prepared for the presentation. Made slides for presentation.
Dec 19th	Finished!

4.9 Project Log

The project log shows the history of all the git commits.

```
commit 6aaa5e8b64d05c10e41adce4ce0613de08c3992f (HEAD -> master, origin/master)
Author: Boya Song <bs3065@columbia.edu>
Date:   Wed Dec 19 22:50:25 2018 -0500

    modify README

commit 5628be0757a043f0cb9419ee9512178297fb9d41
Author: Boya Song <bs3065@columbia.edu>
Date:   Wed Dec 19 10:46:38 2018 -0500
```

```
    small modifications on codegen

commit b1fa985f90714ce0842a84767decae99111e203c
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Wed Dec 19 09:54:15 2018 -0500

    add single line comments, check illegal characters, add test cases

commit 04e24ff40c44d3b6804fad82b3504b3199e7d889
Author: ChunliF <cf2710@columbia.edu>
Date:   Tue Dec 18 23:22:41 2018 -0500

    add semant tests

commit 6402b55adb8c2bc6ddc5316d8e8d50d5ddab04e9
Merge: 05ddf51 f2f16b6
Author: ChunliF <cf2710@columbia.edu>
Date:   Tue Dec 18 21:56:00 2018 -0500

    Merge branch 'master' of https://github.com/Miyeah/PLT-MathLight

commit 05ddf51dfcda2f90ea2a69d056b871b951c32aa7
Author: ChunliF <cf2710@columbia.edu>
Date:   Tue Dec 18 21:55:50 2018 -0500

    non-matrix type not infer size

commit f2f16b64df8a5b9eaf054b4367da5b5ffe161ae3
Author: Boya Song <bs3065@columbia.edu>
Date:   Tue Dec 18 16:27:22 2018 -0500

    Support String

commit a4635af2076f5b1e327db0214cddc09250376caa
Author: Boya Song <bs3065@columbia.edu>
Date:   Tue Dec 18 15:19:28 2018 -0500

    modify demo2

commit 92e83355758073d496e4f54733cdd69d974f0871
Merge: 5f8e125 890476d
Author: ChunliF <cf2710@columbia.edu>
Date:   Tue Dec 18 16:00:14 2018 -0500

    Merge branch 'master' of https://github.com/Miyeah/PLT-MathLight

commit 5f8e12591204c09225b80d6b1664b3c8f05088fc
Author: ChunliF <cf2710@columbia.edu>
Date:   Tue Dec 18 16:00:00 2018 -0500

    fix

commit 890476d924af888bbdf6d2fc135191206d377027
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Tue Dec 18 15:45:58 2018 -0500
```

```
    minor

commit 3824eb6b8a7640cac716e1a22c677cfe0bda21f8
Author: ChunliF <cf2710@columbia.edu>
Date:   Tue Dec 18 15:35:36 2018 -0500

    comment

commit 8e84380ddedb74881205a3e38919d2fe36fe3e42
Author: ChunliF <cf2710@columbia.edu>
Date:   Tue Dec 18 14:12:50 2018 -0500

    add comment

commit 917899661f5b1787d566f5e6ca2b7196ea1386a7
Author: Boya Song <bs3065@columbia.edu>
Date:   Tue Dec 18 13:13:40 2018 -0500

    codegen warning fix

commit 016b1a3bb2c52b5a76c636c1dbd3a6345df5eb5e
Author: ChunliF <cf2710@columbia.edu>
Date:   Tue Dec 18 13:10:52 2018 -0500

    add test script

commit fa732e80235ae3da1f3c5ee7166d7c5d87c8a269
Author: Boya Song <bs3065@columbia.edu>
Date:   Mon Dec 17 20:18:25 2018 -0500

    add test case for pow

commit 3b34b7978864954e5dfcbb371b60579fd8a12106
Author: Boya Song <bs3065@columbia.edu>
Date:   Mon Dec 17 18:02:35 2018 -0500

    global declare and assign works

commit 384b7374eec121f233ee6a7ed4e88b0af17eba0a
Merge: 4478ea6 2a4b5b8
Author: ChunliF <cf2710@columbia.edu>
Date:   Mon Dec 17 17:34:57 2018 -0500

    Merge branch 'master' of https://github.com/Miyeah/PLT-MathLight

commit 4478ea632e1325b8bf3c8a85563eea7aa804d1d5
Author: ChunliF <cf2710@columbia.edu>
Date:   Mon Dec 17 17:34:51 2018 -0500

    pow

commit 2a4b5b885364877c8fc81c503ab4f5cd7c16979f
Author: ChunliF <cf2710@columbia.edu>
Date:   Mon Dec 17 17:26:58 2018 -0500
```

```
    small change

commit a092513a6019cf76a76b50483506bd2712a20ce8
Author: ChunliF <cf2710@columbia.edu>
Date:   Mon Dec 17 17:19:04 2018 -0500

    fix

commit 4ccb9d451c7fe76d1643706a5b59fb840cfeefb6
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Mon Dec 17 17:15:11 2018 -0500

    remove func return size

commit e412aa96df13eb09c00f2c0eeb63bb4456f6327e
Merge: 311d878 ca42579
Author: ChunliF <cf2710@columbia.edu>
Date:   Mon Dec 17 17:10:43 2018 -0500

    Merge branch 'master' of https://github.com/Miyeah/PLT-MathLight

commit 311d878948edb40b2c3c28ee9c31dde064ac25bb
Author: ChunliF <cf2710@columbia.edu>
Date:   Mon Dec 17 17:10:36 2018 -0500

    fix global vars declare and assign

commit ca425797ae230a22f5e131e539d3e96718d1fc37
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Mon Dec 17 16:41:42 2018 -0500

    minor

commit 2819be9853db77b3597062105 32cf6017a3d141d
Author: Boya Song <bs3065@columbia.edu>
Date:   Mon Dec 17 15:56:38 2018 -0500

    modify some malloc to alloca

commit 36094b1c5cc953aca8d02a17fe034bc625087ca4
Author: Boya Song <bs3065@columbia.edu>
Date:   Mon Dec 17 15:50:11 2018 -0500

    clean repo

commit 1cfc7f788a7613b40a9da9150ac841028d7e97a0
Author: Boya Song <bs3065@columbia.edu>
Date:   Mon Dec 17 15:47:04 2018 -0500

    Some modifications in codegen, remove warnings, add log, abs, and power

commit 13838ef2cbdd713458f05a239b40d4af0038bdec
Author: Yuli Han <yulihan@dyn-209-2-226-156.dyn.columbia.edu>
Date:   Mon Dec 17 14:58:43 2018 -0500

    delete size
```

```
commit 4b901ce554356ce35f71d5b9e3a52de6042eb1e8
Author: Yuli Han <yulihan@dyn-209-2-226-156.dyn.columbia.edu>
Date:   Mon Dec 17 13:45:04 2018 -0500

    add norm2

commit 0c4a2449173944164b330e62098e999095848bd0
Merge: efc3fe0 3908424
Author: Yuli Han <yulihan@dyn-209-2-226-156.dyn.columbia.edu>
Date:   Mon Dec 17 11:53:55 2018 -0500

    resolve conflict

commit efc3fe097eb262441e49cd5dd0912ed55b71a7d3
Author: Yuli Han <yulihan@dyn-209-2-226-156.dyn.columbia.edu>
Date:   Mon Dec 17 11:52:36 2018 -0500

    add max_eigvalue

commit 3908424a5497a7c92d6c97275d875a7884daeb3b
Author: Boya Song <bs3065@columbia.edu>
Date:   Mon Dec 17 11:42:48 2018 -0500

    modify demos using declare&assign

commit c3c4542c805c66a674b845df946608632dc87fec
Author: Boya Song <bs3065@columbia.edu>
Date:   Mon Dec 17 11:25:19 2018 -0500

    matrix without size done

commit d8575d6ed07eb5c05d3d525fa112e7bff94ff2b2
Author: Yuli Han <yulihan@dyn-209-2-226-156.dyn.columbia.edu>
Date:   Mon Dec 17 11:26:57 2018 -0500

    add norm1

commit 11b8b7192412dca965b5eca49790c421d4566e96
Merge: d7cb581 5b18bf3
Author: ChunliF <cf2710@columbia.edu>
Date:   Sun Dec 16 23:28:02 2018 -0500

    semant: infer func return size

commit d7cb581b34027a0355f726f5395a0b863e4c2795
Author: ChunliF <cf2710@columbia.edu>
Date:   Sun Dec 16 23:25:26 2018 -0500

    semant: infer func return size"

commit 5b18bf36b6c2329fe5b419882c5cdd5114342660
Author: Boya Song <bs3065@columbia.edu>
Date:   Sun Dec 16 22:12:45 2018 -0500

    declare and assign done
```

```
commit 7ea8b336366d92166c2e601d8ef6940be567f7aa
Author: ChunliF <cf2710@columbia.edu>
Date:   Sun Dec 16 21:30:55 2018 -0500

    fix declare and assign

commit 567dd23946a702d269b3190447cfc98571ff2d10
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Sun Dec 16 21:20:56 2018 -0500

    fix typo

commit d5f3b80e00d575fb9482b0236f94e5d458ef41c4
Author: Boya Song <bs3065@columbia.edu>
Date:   Sun Dec 16 21:11:29 2018 -0500

    parser error

commit b088facd5b28b712b80faade7ad81e87fbea6ba8
Author: ChunliF <cf2710@columbia.edu>
Date:   Sun Dec 16 18:38:35 2018 -0500

    fix

commit 6545cdfc586bee5148fa7e773ae475d36f9fb98e
Author: ChunliF <cf2710@columbia.edu>
Date:   Sun Dec 16 18:33:21 2018 -0500

    fix

commit 53990ccebde449fc080e98affc017839e06c2d56
Author: ChunliF <cf2710@columbia.edu>
Date:   Sun Dec 16 18:29:40 2018 -0500

    semant: declare and assignment

commit cb523297a39c7bb66e681b725a71ea57acfd5c8a
Merge: 21e3a11 76ecf3a
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Sun Dec 16 17:50:27 2018 -0500

    Merge branch 'master' of https://github.com/Miyeah/PLT-MathLight

commit 21e3a11bfa7bebbc96663daab7aee2eff46080e8
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Sun Dec 16 17:50:25 2018 -0500

    implement declaration&assign

commit 76ecf3adbbe423ad62e18709953c0c4e4a9cd9ad
Author: Yuli Han <yulihan@dyn-209-2-227-72.dyn.columbia.edu>
Date:   Sun Dec 16 16:17:58 2018 -0500

    add test case for 1d element
```

```
commit 6f2854f1bd081bc2f7b2410c21052717ee850223
Merge: 3a05f2e f9bead4
Author: Yuli Han <yulihan@dyn-209-2-227-72.dyn.columbia.edu>
Date:   Sun Dec 16 16:17:01 2018 -0500

    Merge branch 'master' of https://github.com/Miyeah/PLT-MathLight

commit 3a05f2e34f17c449506061fc1182fa9cc76808a5
Author: Yuli Han <yulihan@dyn-209-2-227-72.dyn.columbia.edu>
Date:   Sun Dec 16 16:16:58 2018 -0500

    d

commit f9bead45407aa2bbc228503f852be530f14b838d
Author: ChunliF <cf2710@columbia.edu>
Date:   Sun Dec 16 16:16:16 2018 -0500

    parser 1d assignment 0->1

commit 9f1ad04aa6325307e22ceaa3e4c221e9f04d6580
Author: Yuli Han <yulihan@dyn-209-2-227-72.dyn.columbia.edu>
Date:   Sun Dec 16 15:05:28 2018 -0500

    add demo3

commit 1252d98952e1fb7dbd76d2bed889d6fe26b308f0
Author: Yuli Han <yulihan@dyn-209-2-227-72.dyn.columbia.edu>
Date:   Sun Dec 16 14:43:10 2018 -0500

    add sizeof row and sizeof col

commit 7376a0b4460dd11686af77f121d6ac0e2fddfbc0
Author: Yuli Han <yulihan@dyn-209-2-227-72.dyn.columbia.edu>
Date:   Sun Dec 16 14:17:22 2018 -0500

    add trace

commit ae163d384e5908ab116cc6185c6ef3d55d2b2450
Author: Yuli Han <yulihan@dyn-209-2-227-72.dyn.columbia.edu>
Date:   Sun Dec 16 13:37:23 2018 -0500

    modify demo2

commit 06273b4d66e4933acb37e259db408616734e656f
Author: Yuli Han <yulihan@dyn-209-2-227-72.dyn.columbia.edu>
Date:   Sun Dec 16 13:36:45 2018 -0500

    modify demo 1

commit 8a3529a3de7d941da45a8e887c56ee20c803abb6
Author: Yuli Han <yulihan@dyn-209-2-227-72.dyn.columbia.edu>
Date:   Sun Dec 16 13:34:00 2018 -0500

    remove min

commit 2e0a82e3383746eee2444a9db1fe6213b4219d1e
```

```
Merge: 0dccd65 915f41c
Author: Yuli Han <yulihan@dyn-209-2-227-72.dyn.columbia.edu>
Date:   Sun Dec 16 13:21:44 2018 -0500

    merge

commit 0dccd659892825810f424131e7838a13dd2bd459
Author: Yuli Han <yulihan@dyn-209-2-227-72.dyn.columbia.edu>
Date:   Sun Dec 16 13:20:31 2018 -0500

    add double casting

commit 915f41c2d77dcac10b4129d2a5d71bc7902e016d
Author: Boya Song <bs3065@columbia.edu>
Date:   Sun Dec 16 13:16:30 2018 -0500

    matrix element change done

commit ea3ed151a41dce717631d31656b994fb8696a9df
Author: ChunliF <cf2710@columbia.edu>
Date:   Sun Dec 16 12:51:36 2018 -0500

    semant: add matrix modify

commit 069a742d5dd5ba0b3feb433c4ba7449b97a4810d
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Sun Dec 16 02:28:13 2018 -0500

    support negative matrix primitive

commit 5b2515c7d5143f7ca543fdbc708f24c8d5d4568f
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Sun Dec 16 01:51:17 2018 -0500

    check that only matrix type can declare size, fix test cases

commit 2249f15bdf1c878f44504a912bf6f614920d1d19
Merge: 3d63ff6 0e0b05e
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Sat Dec 15 22:51:10 2018 -0500

    Merge branch 'master' of https://github.com/Miyeah/PLT-MathLight

commit 3d63ff6e2d993fbcb8efd5ffeba58d5dc223a299
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Sat Dec 15 22:50:58 2018 -0500

    parse Matrix1D/2DModify

commit 0e0b05e3be748431c13ae5faa23c4ab4ab76ad34
Author: Boya Song <bs3065@columbia.edu>
Date:   Sat Dec 15 20:48:20 2018 -0500

    demo2 complete and make demo1 becomes longer.

commit 82aa0caf0ccfd12d82082240474735e6b0562060
```

```
Author: Yuli Han <yulihan@YulideMacBook-Air.local>
Date:   Sat Dec 15 19:40:30 2018 -0500

    first demo pass

commit 72e3cbf3d4dc014bc198d9b063f79dbb8e7b39bb
Merge: 57de364 dcec08e
Author: Yuli Han <yulihan@YulideMacBook-Air.local>
Date:   Sat Dec 15 19:32:38 2018 -0500

    d

commit 57de3649c76ec82c1aa34bf3a9215fa1acc6e4fc
Author: Yuli Han <yulihan@YulideMacBook-Air.local>
Date:   Sat Dec 15 19:31:49 2018 -0500

    update test

commit dcec08e028847b7878b9674272dd403f1c48a40e
Author: Boya Song <bs3065@columbia.edu>
Date:   Sat Dec 15 19:30:54 2018 -0500

    matrix return works

commit 36abde11ed9c3e5d6388b6a157a163f6929b8b9e
Author: Yuli Han <yulihan@dyn-209-2-227-72.dyn.columbia.edu>
Date:   Sat Dec 15 17:08:43 2018 -0500

    matrix add double

commit be4393dcc4555b8a66018311c527d59815339918
Author: Yuli Han <yulihan@dyn-209-2-227-72.dyn.columbia.edu>
Date:   Sat Dec 15 17:06:32 2018 -0500

    double add matrix

commit 8c8c8d551f670ff3d788e38f431513335880f99c
Author: Boya Song <bs3065@columbia.edu>
Date:   Sat Dec 15 16:48:36 2018 -0500

    linear regression test case pass

commit df197fe847a853a75f52f86b323c11358833add0
Merge: 23b0332 78c7927
Author: ChunliF <cf2710@columbia.edu>
Date:   Sat Dec 15 16:27:22 2018 -0500

    merge

commit 23b0332121d72ac7629f4c11b69aa683ce842631
Author: ChunliF <cf2710@columbia.edu>
Date:   Sat Dec 15 16:26:33 2018 -0500

    semant, add data_size for func

commit 78c7927ea716c2a4e79d1e6e88ef65511e2c23b7
```

```
Merge: 4643c09 46dbd25
Author: Yuli Han <yulihan@dyn-209-2-227-72.dyn.columbia.edu>
Date:   Sat Dec 15 16:24:40 2018 -0500

    merge

commit 4643c095e62bb1ee42fda14f1b8afa32f252d26f
Author: Yuli Han <yulihan@dyn-209-2-227-72.dyn.columbia.edu>
Date:   Sat Dec 15 16:23:31 2018 -0500

    add sum and mean

commit 46dbd25cfeb58a2f3f2fb477cca42ecd8768dc35
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Sat Dec 15 16:23:30 2018 -0500

    parse function retrun type's size

commit 71751ded9f5091389 41f8f3170f0860119bad98c
Author: Boya Song <bs3065@columbia.edu>
Date:   Sat Dec 15 14:31:11 2018 -0500

    add inv and modify the implementation for determinant

commit b2b0c3eae57b5cf686a2ce916ef7d951c92fe78f
Merge: b340a62 0ee3328
Author: ChunliF <cf2710@columbia.edu>
Date:   Sat Dec 15 11:59:28 2018 -0500

    merge

commit b340a623cb51369bd34e4af24713e40ebc5ca7f8
Author: ChunliF <cf2710@columbia.edu>
Date:   Sat Dec 15 11:57:26 2018 -0500

    semant: add number type conversion"

commit 0ee3328efafef100c60dccd7db8a84cb2f6a5599
Merge: b61b739 3b1e75d
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Fri Dec 14 21:08:50 2018 -0500

    Merge branch 'master' of https://github.com/Miyeah/PLT-MathLight

commit b61b739b107b56c604734f91646 98a0787acf69d
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Fri Dec 14 21:08:47 2018 -0500

    add size declaration for formal list, fix one dim size, auto convert int to double in matrix
declaration

    1. add size declaration for formal list
    2. fix one dim size issue
    3. auto convert int to double in matrix declaration

commit 3b1e75d2b51a4b1e5ad11a6db965e7a167e1e2d8
```

```
Author: Yuli Han <yulihan@YulideMacBook-Air.local>
Date:   Fri Dec 14 20:07:12 2018 -0500

    eigvalue todo

commit b8c83d39454 51e126f5804dc641a8863754bd855
Author: Boya Song <bs3065@columbia.edu>
Date:   Fri Dec 14 19:23:40 2018 -0500

    link to det works

commit f4946d6ff3f15b262af2e9c5218241711f996f3ab
Author: ChunliF <cf2710@columbia.edu>
Date:   Fri Dec 14 17:44:20 2018 -0500

    initial size -> (-1, -1)

commit 3614b5020f7dd07a6cc8e85102fc0e9335e75900
Author: Yuli Han <yulihan@YulideMacBook-Air.local>
Date:   Fri Dec 14 17:08:09 2018 -0500

    add link

commit fb22bd038cd53026b03d61416c11a27ffcea7d8a
Author: Boya Song <bs3065@columbia.edu>
Date:   Fri Dec 14 13:55:22 2018 -0500

    linking in process

commit d5c51bdc89d719b4c554966611ba4b6bab942c2a
Author: Yuli Han <yulihan@YulideMacBook-Air.local>
Date:   Fri Dec 14 13:27:09 2018 -0500

    add matrix dot operation

commit 178b03fdee151a2aafacdfd2bead84c018fbd864
Author: Yuli Han <yulihan@YulideMacBook-Air.local>
Date:   Fri Dec 14 11:50:54 2018 -0500

    add matrix sizeof

commit bbcafe2826126f15901591906be8d4d1c61aeb93
Author: Yuli Han <yulihan@YulideMacBook-Air.local>
Date:   Fri Dec 14 11:45:57 2018 -0500

    add control flow

commit 173fc8559b9a732c42105924f3ecc64bacb850df
Merge: bc7d58f 6c8ba04
Author: ChunliF <cf2710@columbia.edu>
Date:   Thu Dec 13 20:42:50 2018 -0500

    Merge branch 'master' of https://github.com/Miyeah/PLT-MathLight

commit bc7d58f86ddf9c1af95741 67a39e70a878774e62
Author: ChunliF <cf2710@columbia.edu>
```

```
Date:   Thu Dec 13 20:40:15 2018 -0500

    semant: add fail tests

commit 6c8ba04db40dad31c1d222733051b66c7e23fd8f
Author: Boya Song <bs3065@columbia.edu>
Date:   Thu Dec 13 20:28:43 2018 -0500

    Add unop including transpose but abs.

commit 4ffad9a2cc204f9bf92f2639aed4cb518de69ca7
Author: Yuli Han <yulihan@dyn-209-2-226-103.dyn.columbia.edu>
Date:   Thu Dec 13 20:09:04 2018 -0500

    add matrix mul double

commit 800223e43310f7eaf18fad417afd1f02a30e2bcb
Author: Yuli Han <yulihan@dyn-209-2-226-103.dyn.columbia.edu>
Date:   Thu Dec 13 18:18:34 2018 -0500

    fix matrix concat

commit c17c8d20fd625cc79e10b8f15db9f3e52afece77
Author: Yuli Han <yulihan@dyn-209-2-226-103.dyn.columbia.edu>
Date:   Thu Dec 13 18:02:52 2018 -0500

    fix matrix operation issues

commit 1ac0ba0f2bde60ecfbb66c6e312e4fd34aa0b174
Author: Yuli Han <yulihan@dyn-209-2-226-103.dyn.columbia.edu>
Date:   Thu Dec 13 18:02:36 2018 -0500

    fix matrix ope issues

commit ac9becb031e6869f117f25ac6660154e9ce47ae7
Merge: f45d31f 0025d7d
Author: Yuli Han <yulihan@dyn-209-2-226-103.dyn.columbia.edu>
Date:   Thu Dec 13 17:46:40 2018 -0500

    Merge branch 'master' of https://github.com/Miyeah/PLT-MathLight into temp

commit f45d31fccb6c7b23c2dc7a4782a28e39b7d50a86
Author: Yuli Han <yulihan@dyn-209-2-226-103.dyn.columbia.edu>
Date:   Thu Dec 13 17:43:15 2018 -0500

    fix

commit 72411913d8022cfeff984001eb5ec963108eb440
Author: Yuli Han <yulihan@dyn-209-2-226-103.dyn.columbia.edu>
Date:   Thu Dec 13 17:43:05 2018 -0500

    fix matrix add

commit 0025d7d09398c71464d53c43d3cf2db0e620fc9d
Author: Boya Song <bs3065@columbia.edu>
Date:   Thu Dec 13 17:20:19 2018 -0500
```

```
    fix built-in function fill

commit 8272a490df79ceaf0e68139a99df3cc7c38845a7
Author: Boya Song <bs3065@columbia.edu>
Date:   Thu Dec 13 17:08:00 2018 -0500

    change print implementation so that it can support string and matrix

commit 0ff7b550518d782360d5f72d8e70817d45efe8cc
Author: Boya Song <bs3065@columbia.edu>
Date:   Thu Dec 13 13:28:34 2018 -0500

    pointer to matrix works

commit a4cd2daadc8e3462e234195dedbd8458c639c886
Author: Boya Song <bs3065@columbia.edu>
Date:   Thu Dec 13 13:43:40 2018 -0500

    test matrix access

commit dc553191fc4d5c8787ed81e87d0c27c24a4eaa18
Author: ChunliF <cf2710@columbia.edu>
Date:   Thu Dec 13 13:17:21 2018 -0500

    add matrix elementwise operator

commit ee9b7eb74d033d2f71658dd8a2248b9525621955f
Merge: c921277 7563733
Author: ChunliF <cf2710@columbia.edu>
Date:   Thu Dec 13 12:41:46 2018 -0500

    Merge branch 'master' of https://github.com/Miyeah/PLT-MathLight

commit c9212777c4b1e468d552ed866045969fa43b03a3
Author: ChunliF <cf2710@columbia.edu>
Date:   Thu Dec 13 12:40:43 2018 -0500

    add buildin func

commit 7563733df45bfd01c3f4fa57d4ecc28c2de58c26
Merge: 80ae24f 7021070
Author: Yuli Han <yulihan@dyn-209-2-226-103.dyn.columbia.edu>
Date:   Thu Dec 13 11:22:12 2018 -0500

    fix conflict

commit 80ae24f6dff2394a92845b0e29ecf06216bc54e7
Author: Yuli Han <yulihan@dyn-209-2-226-103.dyn.columbia.edu>
Date:   Thu Dec 13 11:20:50 2018 -0500

    update matrix mul

commit f2ac4f2d3d0fed372c44311f635bbb1a0286b4ba
Author: Yuli Han <yulihan@dyn-209-2-226-103.dyn.columbia.edu>
Date:   Thu Dec 13 11:20:42 2018 -0500
```

```
    update matrix mul

commit 70210705e7d6c099c4f85134d2caef73d83c8736
Author: Boya Song <bs3065@columbia.edu>
Date:   Thu Dec 13 11:07:06 2018 -0500

    add function return

commit 9563dba2c7400bee03e899712410800c1dd944450
Merge: 1ae1683 8929f1a
Author: Yuli Han <yulihan@dyn-209-2-226-103.dyn.columbia.edu>
Date:   Thu Dec 13 10:50:36 2018 -0500

    add concat

commit 1ae16839993be65b07357066e07e48df54181e97
Author: Yuli Han <yulihan@dyn-209-2-226-103.dyn.columbia.edu>
Date:   Thu Dec 13 10:50:13 2018 -0500

    add concat

commit 8929f1a632926f77744b7b256970800717df7256
Author: Boya Song <bs3065@columbia.edu>
Date:   Wed Dec 12 19:49:49 2018 -0500

    Add codegen for 1D/2D matrix access

commit 4d7eaa953b5e2bef22506b298eb06ce3142f5a0d
Merge: 2a37674 6f30ffc
Author: ChunliF <cf2710@columbia.edu>
Date:   Wed Dec 12 17:51:12 2018 -0500

    Merge branch 'master' of https://github.com/Miyeah/PLT-MathLight

commit 2a376749675619c56dffec09c9ef7953eb405367
Author: ChunliF <cf2710@columbia.edu>
Date:   Wed Dec 12 17:51:05 2018 -0500

    semant: add range

commit 6f30ffc0333a89034a115078502277a36f747ed4
Author: Boya Song <bs3065@columbia.edu>
Date:   Wed Dec 12 17:41:06 2018 -0500

    sqrt test fixed

commit f5a41a77f81579469d5d5e26812438adf79b4b6d
Author: Boya Song <bs3065@columbia.edu>
Date:   Wed Dec 12 17:36:43 2018 -0500

    add sqrt and fill

commit 1be40e53a00ca960a390e32cefab9dc2611bfa4d
Merge: 3467215 a29a465
Author: Yuli Han <yulihan@dyn-209-2-227-120.dyn.columbia.edu>
```

```
Date:   Wed Dec 12 17:20:41 2018 -0500

    fix conflict

commit 346721517699df118914d9c2a974d0bd58dd407f2
Author: Yuli Han <yulihan@dyn-209-2-227-120.dyn.columbia.edu>
Date:   Wed Dec 12 17:19:39 2018 -0500

    add ope

commit a29a465cb83543e1c1983927938a03e790dee4178
Author: ChunliF <cf2710@columbia.edu>
Date:   Wed Dec 12 16:22:42 2018 -0500

    semant: add matrix index

commit 7a9c3b06b6b2bea9218510766bbdb51ed0d1ff0e0
Author: ChunliF <cf2710@columbia.edu>
Date:   Wed Dec 12 16:09:50 2018 -0500

    semant: add built func

commit 2b99514d4270f37d054cbadd8252a07f6b0d5369
Author: ChunliF <cf2710@columbia.edu>
Date:   Wed Dec 12 15:54:04 2018 -0500

    semant: add matrix support, add uop binop

commit 2afc650026de2d933d5e006620fc16ed43ebdc77
Merge: dd000d6 3577ec1
Author: ChunliF <cf2710@columbia.edu>
Date:   Wed Dec 12 15:47:10 2018 -0500

    Merge branch 'master' of https://github.com/Miyeah/PLT-MathLight

commit dd000d6468a37734b5c70cc3d8f13e3091a9bc0b
Author: ChunliF <cf2710@columbia.edu>
Date:   Wed Dec 12 15:46:51 2018 -0500

    semant add matrix support, add matrix binop

commit 3577ec16ec002b60ca73d460ad34c3f6d0dcbe8c
Author: Boya Song <bs3065@columbia.edu>
Date:   Wed Dec 12 15:18:29 2018 -0500

    Fix small issue in building the matrix literal

commit 0bc40e4279fcb3d90062e7629cac41ad2ad610891
Merge: 151bcc2 2f0cb72
Author: Yuli Han <yulihan@dyn-209-2-227-120.dyn.columbia.edu>
Date:   Wed Dec 12 14:30:38 2018 -0500

    Merge branch 'master' of https://github.com/Miyeah/PLT-MathLight into add-ope

commit 151bcc2eba97c34b06ba7abab2f2c39084488bd55
Author: Yuli Han <yulihan@dyn-209-2-227-120.dyn.columbia.edu>
```

```
Date:   Wed Dec 12 14:26:29 2018 -0500

    add int operator

commit 2f0cb722cbec7ed69ff688c2f02b330713f899f4
Author: Boya Song <bs3065@columbia.edu>
Date:   Wed Dec 12 14:25:50 2018 -0500

    assign works except matrix

commit 764f67ad0238ce9b6816ce7c2b9689a482783d5d
Author: Boya Song <bs3065@columbia.edu>
Date:   Wed Dec 12 11:16:31 2018 -0500

    fix codegen and semant for matrix declaration with size

commit 66fe300d3973921761534416297de174dbfa7256
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Sat Dec 8 03:11:14 2018 -0500

    add declaration for matrix size

commit 998e0de9520cb667ce48825d3a003172edf21cd5
Author: Boya Song <bs3065@columbia.edu>
Date:   Fri Dec 7 13:13:57 2018 -0500

    Fix print so that it can now support any expression and add matrix representation

commit cf43e41cab11b2923fe7f0e406f80001af458642
Merge: d4a31ff c51cf74
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Thu Dec 6 18:55:45 2018 -0500

    Merge branch 'master' of https://github.com/Miyeah/PLT-MathLight

commit d4a31ff502a2ec29e2abefc572867089066d5888c
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Thu Dec 6 18:55:41 2018 -0500

    Fix MatrixOperation

commit c51cf74f18059da6738e144687a1e8bce3ca81c4
Author: Boya Song <bs3065@columbia.edu>
Date:   Thu Dec 6 17:45:24 2018 -0500

    modify the test case for matrix to be 2D

commit 9be4a466af3226fb9963cf8393f72b97860499a1a
Author: Boya Song <bs3065@columbia.edu>
Date:   Thu Dec 6 15:33:28 2018 -0500

    Change top level file to be able to print AST and SAST

commit 248cb0be7a23141fc01d4196421081d7505ac855
Author: ChunliF <cf2710@columbia.edu>
Date:   Thu Dec 6 15:20:13 2018 -0500
```

```
    add pretty print, add other type support for print

commit 18327f67806a99e6c18e74e7ea8a0155175b551f
Merge: 97010a1 a0a6862
Author: ChunliF <cf2710@columbia.edu>
Date:   Thu Dec 6 15:07:08 2018 -0500

    Merge branch 'master' of https://github.com/Miyeah/PLT-MathLight

commit 97010a1a61ec39984753f822486979 61ae233ee8
Author: ChunliF <cf2710@columbia.edu>
Date:   Thu Dec 6 15:06:06 2018 -0500

    add pretty print + support for print(other than string)"

commit a0a68625ed462365f26734bc4a2a55ccd7ff602f
Author: Boya Song <bs3065@columbia.edu>
Date:   Thu Dec 6 14:56:44 2018 -0500

    [Code generation for print]
    1. Support print for all types
    2. Add test cases for each type
    PS: Matrix should be printed like this, e.g. for [1.0,2.0,3.0;1.1,1.2,1.3] is printed
    1.0 2.0 3.0
    1.1 1.2 1.3

commit bc94fafa263dd580300ecb8be10057cf0bbf4278
Author: Boya Song <bs3065@columbia.edu>
Date:   Tue Dec 4 12:29:46 2018 -0500

    one more test case about variable

commit dcae9675dc99318baa2aad8f043b2237c1288ee0
Author: Boya Song <bs3065@columbia.edu>
Date:   Wed Nov 14 20:35:04 2018 -0500

    Modify Readme

commit c001453fdc69740e3bc44fa55afd4b057e974e05
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Wed Nov 14 13:23:55 2018 -0500

    parser comments

commit 21948db6d045bce1de83624d143e721e0fcf759f
Author: Boya Song <bs3065@columbia.edu>
Date:   Tue Nov 13 20:16:24 2018 -0500

    add merlin in gitignore

commit 92c1cea9a41d5d25fbb22f418b8ec27674a4ce21
Author: Boya Song <bs3065@columbia.edu>
Date:   Mon Nov 12 16:47:05 2018 -0500

    Hello World
```

```
commit 3f8efa267e84d17c31e2206c1a0790c90e4459c7
Author: Boya Song <bs3065@columbia.edu>
Date:   Mon Nov 12 13:56:50 2018 -0500

    fixes in codegen and testall, also add a failure case

commit 2a78856dd9fdbe2aa33f403d37439d2a9128e762
Author: Miyeah Chen <fzcmy@qq.com>
Date:   Mon Nov 12 13:12:32 2018 -0500

    fix parser&ast

commit 83495a5fc15a4a61a62caa732717bfbd73132987
Author: Boya Song <bs3065@columbia.edu>
Date:   Mon Nov 12 09:35:48 2018 -0500

    Rename ast file and changes in testall.sh

commit 850f9dc1070635959e7d2091694dc98f1dab23cb
Author: Boya Song <bs3065@columbia.edu>
Date:   Sun Nov 11 21:40:34 2018 -0500

    fix mathlight warnings

commit 33e6c5fef4701b5e41aac9eec15303c16c629967
Author: ChunliF <cf2710@columbia.edu>
Date:   Sun Nov 11 21:37:54 2018 -0500

    fix semant

commit a7b62919511cecaccde6c6780119533bce8fee4d4c
Author: Boya Song <bs3065@columbia.edu>
Date:   Sun Nov 11 21:09:47 2018 -0500

    put building files into gitignore

commit 8522efa0ebd5bea54c5c68dc53104130abb87063
Author: Boya Song <bs3065@columbia.edu>
Date:   Sun Nov 11 21:08:00 2018 -0500

    small fix on codegen.ml

commit 3c24041d4604cc35456ffcaeabaab81206177cced
Author: Boya Song <bs3065@columbia.edu>
Date:   Sun Nov 11 20:59:01 2018 -0500

    remove gitignore

commit f160b9b7f4d12ead3a008f747344a166f365800fb
Author: ChunliF <cf2710@columbia.edu>
Date:   Sun Nov 11 16:07:42 2018 -0500

    semant add print support

commit c813a632913ba54be8c6e2c327d313304ef334ed
```

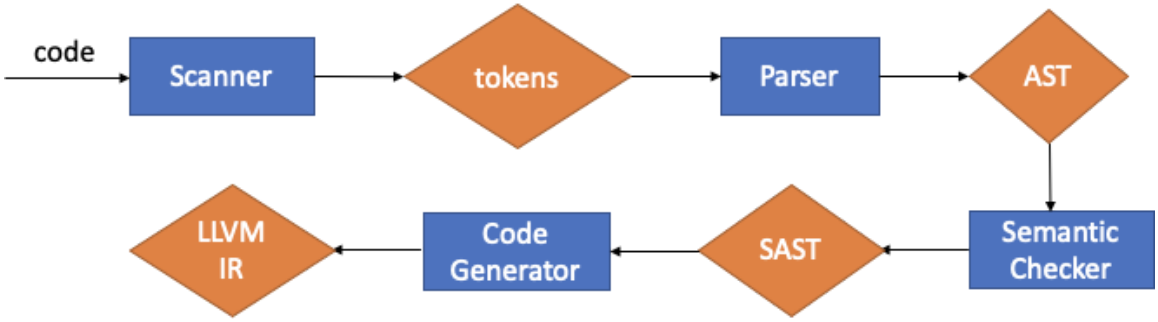
```
Author: Boya Song <bs3065@columbia.edu>
Date:   Sun Nov 11 14:18:17 2018 -0500

    Add codegen, top level, the test file, makefile etc for HelloWorld

commit 8698f8a5f5c6f162c7e3bb5fa1727aae84f1c026
Author: ChunliF <cf2710@columbia.edu>
Date:   Sun Nov 11 14:03:46 2018 -0500

    empty semantic check

commit be5d9f8dfeaf892d5db2d86ff48e3a27c583cea4
Author: Boya Song <bs3065@columbia.edu>
Date:   Mon Oct 15 20:53:50 2018 -0400

    Add '//' comment and small modification for 2d array initialization

commit 0fa3dca498fd38651821b63e41e1a0b21de08d90
Author: Boya Song <bs3065@columbia.edu>
Date:   Mon Oct 15 19:59:49 2018 -0400

    MatrixLit small modification

commit 978df77395e9ab936baa5850bdf38b080816fa82
Author: Boya Song <bs3065@columbia.edu>
Date:   Mon Oct 15 15:10:16 2018 -0400

    Use comma for Horizontal concat and semicolon for Vertical Concat. Not so sure if it is correct or
not. but the parsing works fine

commit 375e6eedf2a2bae5eb2fea76ade3a4793270b2c2
Author: Boya Song <bs3065@columbia.edu>
Date:   Mon Oct 15 11:56:19 2018 -0400

    Parser Updated with 2D matrix, but currently not check whether dimensions match

commit fdbd0429df3b4b383ad0cbf3ea4736660a45d05d
Author: Boya Song <bs3065@columbia.edu>
Date:   Mon Oct 15 11:46:21 2018 -0400

    Initial Commit. Scanner and AST by Yuli, and Parser, example by Mingye.
```

5. Architectural Design

5.1 Architecture Overview

The architecture of Mathlight consists of four parts: scanner, parser, semantic checker and code generator, shown in the figure below. The scanner, parser and semantic checker are the frontend of the compiler and the code generator is the backend.

The entry point of mathlight is mathlight.ml, which calls all of these 4 components sequentially. Firstly, the code is passed into scanner and parser to generate AST. Then semantic checker takes AST as input and then checks it using post-order traversal and generate SAST. Finally, SAST is passed into code generator to get LLVM IR.



5.2 Scanner (Mingye, Yuli)

The scanner is written based on *ocamllex*, which produces a lexical analyzer from a set of regular expressions. The lexer takes the input MathLight source program then produces tokens for keywords, identifiers, operators, literals and constants. If input characters cannot be processed, scanner will report the illegal character error.

5.3 Parser (Mingye)

The parser is written based on *ocamlyacc*, which produces a parser from a context-free grammar. Parser takes generated tokens from scanner as input, and then generates the abstract syntax tree (AST) for further use based on MathLight syntax.

5.4 Semantic Checker (Chunli)

Semantic Checker takes AST generated from parser as input, and then traverse it using post order traversal. When processing every AST node, it does semantic checking and

generate corresponding node for SAST, which extends AST with type information. We build symbol tables and function declaration maps to save type and size for further checking use. Finally, it generates SAST and passes it to code generator.

5.5 Code Generator (Boya, Yuli)

The Code generator is the backend implementation of the compiler of our language. It takes the SAST generated from semantic checker as input, output the LLVM IR. For each SAST node, we built LLVM basic blocks and control flow graph and translated the node into LLVM intermediate representation.

We also linked a C object file. We wrote the function we needed in C language, then compiled it to an object file and called the corresponding C function in codegen.ml file.

6. Test Plan

6.1 Representative programs

1. Analytical method for Linear regression source program:

```
func int main() {
  matrix x <3, 1> = [1;2;3];
  matrix y <3, 1> = [4;5;6];
  matrix ones <3, 1> = fill(3, 1, 1.0);
  matrix x_new <3, 2>;
  matrix w <2, 1>;
  matrix y_lr <3, 1>;

  /* analytical solution for linear regression */
  /* x_pre * w_pre + b = y
     [x_pre : ones] * w = y
     x * w = y
     x' * x * w = x' * y
     w = inv(x' * x) * x' * y */
  x_new = [x, ones];
  w = inv(x_new' * x_new) * x_new' * y;
  y_lr = x_new * w;

  print(w);
  print(y_lr);

  return 0;
```

```
}
```

Target program:

```
; ModuleID = 'MathLight'
source_filename = "MathLight"

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.1 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt.2 = private unnamed_addr constant [4 x i8] c"%g\0A\00"
@fmt.3 = private unnamed_addr constant [4 x i8] c"%g \00"
@fmt.4 = private unnamed_addr constant [2 x i8] c"\0A\00"

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

declare double @sqrt(double)

declare i32 @abs(i32)

declare double @fabs(double)

declare double @pow(double, double)

declare double @log(double)

define i32 @main() {
entry:
  %x = alloca [3 x [1 x double]]*
  %malloccall = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (double* getelementptr
(double, double* null, i32 1) to i64), i64 3) to i32))
  %res = bitcast i8* %malloccall to [3 x [1 x double]]*
  %tmp = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res, i32 0, i32 0, i32 0
  store double 1.000000e+00, double* %tmp
  %tmp1 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res, i32 0, i32 1, i32 0
  store double 2.000000e+00, double* %tmp1
  %tmp2 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res, i32 0, i32 2, i32 0
  store double 3.000000e+00, double* %tmp2
  store [3 x [1 x double]]* %res, [3 x [1 x double]]** %x
  %y = alloca [3 x [1 x double]]*
  %malloccall3 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (double* getelementptr
(double, double* null, i32 1) to i64), i64 3) to i32))
  %res4 = bitcast i8* %malloccall3 to [3 x [1 x double]]*
  %tmp5 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res4, i32 0, i32 0, i32 0
  store double 4.000000e+00, double* %tmp5
  %tmp6 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res4, i32 0, i32 1, i32 0
  store double 5.000000e+00, double* %tmp6
  %tmp7 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res4, i32 0, i32 2, i32 0
  store double 6.000000e+00, double* %tmp7
  store [3 x [1 x double]]* %res4, [3 x [1 x double]]** %y
  %ones = alloca [3 x [1 x double]]*
  %malloccall8 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (double* getelementptr
```

```
(double, double* null, i32 1) to i64), i64 3) to i32))
  %res9 = bitcast i8* %malloccall18 to [3 x [1 x double]]*
  %tmp10 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res9, i32 0, i32 0, i32 0
  store double 1.000000e+00, double* %tmp10
  %tmp11 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res9, i32 0, i32 1, i32 0
  store double 1.000000e+00, double* %tmp11
  %tmp12 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res9, i32 0, i32 2, i32 0
  store double 1.000000e+00, double* %tmp12
  store [3 x [1 x double]]* %res9, [3 x [1 x double]]** %ones
  %x_new = alloca [3 x [2 x double]]*
  %w = alloca [2 x [1 x double]]*
  %y_lr = alloca [3 x [1 x double]]*
  %x13 = load [3 x [1 x double]]*, [3 x [1 x double]]** %x
  %ones14 = load [3 x [1 x double]]*, [3 x [1 x double]]** %ones
  %tmp15 = load [3 x [1 x double]], [3 x [1 x double]]* %x13
  %tmp16 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %x13, i32 0, i32 0
  %indexing = load [1 x double], [1 x double]* %tmp16
  %tmp17 = load [3 x [1 x double]], [3 x [1 x double]]* %ones14
  %tmp18 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %ones14, i32 0, i32 0
  %indexing19 = load [1 x double], [1 x double]* %tmp18
  %malloccall20 = tail call i8* @malloc(i32 trunc (i64 mul (i64 ptrtoint (double* getelementptr (double,
double* null, i32 1) to i64), i64 6) to i32))
  %res21 = bitcast i8* %malloccall20 to [3 x [2 x double]]*
  %tmp22 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %res21, i32 0, i32 0, i32 0
  %tmp23 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %x13, i32 0, i32 0, i32 0
  %tmp24 = load double, double* %tmp23
  store double %tmp24, double* %tmp22
  %tmp25 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %res21, i32 0, i32 0, i32 1
  %tmp26 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %ones14, i32 0, i32 0, i32 0
  %tmp27 = load double, double* %tmp26
  store double %tmp27, double* %tmp25
  %tmp28 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %res21, i32 0, i32 1, i32 0
  %tmp29 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %x13, i32 0, i32 1, i32 0
  %tmp30 = load double, double* %tmp29
  store double %tmp30, double* %tmp28
  %tmp31 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %res21, i32 0, i32 1, i32 1
  %tmp32 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %ones14, i32 0, i32 1, i32 0
  %tmp33 = load double, double* %tmp32
  store double %tmp33, double* %tmp31
  %tmp34 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %res21, i32 0, i32 2, i32 0
  %tmp35 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %x13, i32 0, i32 2, i32 0
  %tmp36 = load double, double* %tmp35
  store double %tmp36, double* %tmp34
  %tmp37 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %res21, i32 0, i32 2, i32 1
  %tmp38 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %ones14, i32 0, i32 2, i32 0
  %tmp39 = load double, double* %tmp38
  store double %tmp39, double* %tmp37
  store [3 x [2 x double]]* %res21, [3 x [2 x double]]** %x_new
  %x_new40 = load [3 x [2 x double]]*, [3 x [2 x double]]** %x_new
  %tmp41 = load [3 x [2 x double]], [3 x [2 x double]]* %x_new40
  %tmp42 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new40, i32 0, i32 0
  %indexing43 = load [2 x double], [2 x double]* %tmp42
  %malloccall44 = tail call i8* @malloc(i32 trunc (i64 mul (i64 ptrtoint (double* getelementptr (double,
double* null, i32 1) to i64), i64 6) to i32))
  %res45 = bitcast i8* %malloccall44 to [2 x [3 x double]]*
  %tmp46 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 0, i32 0
```

```
  %tmp47 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new40, i32 0, i32 0, i32 0
  %tmp48 = load double, double* %tmp47
  store double %tmp48, double* %tmp46
  %tmp49 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 0, i32 1
  %tmp50 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new40, i32 0, i32 1, i32 0
  %tmp51 = load double, double* %tmp50
  store double %tmp51, double* %tmp49
  %tmp52 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 0, i32 2
  %tmp53 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new40, i32 0, i32 2, i32 0
  %tmp54 = load double, double* %tmp53
  store double %tmp54, double* %tmp52
  %tmp55 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 1, i32 0
  %tmp56 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new40, i32 0, i32 0, i32 1
  %tmp57 = load double, double* %tmp56
  store double %tmp57, double* %tmp55
  %tmp58 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 1, i32 1
  %tmp59 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new40, i32 0, i32 1, i32 1
  %tmp60 = load double, double* %tmp59
  store double %tmp60, double* %tmp58
  %tmp61 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 1, i32 2
  %tmp62 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new40, i32 0, i32 2, i32 1
  %tmp63 = load double, double* %tmp62
  store double %tmp63, double* %tmp61
  %x_new64 = load [3 x [2 x double]]*, [3 x [2 x double]]** %x_new
  %tmp65 = load [2 x [3 x double]], [2 x [3 x double]]* %res45
  %tmp66 = load [3 x [2 x double]], [3 x [2 x double]]* %x_new64
  %tmp67 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new64, i32 0, i32 0
  %indexing68 = load [2 x double], [2 x double]* %tmp67
  %malloccall69 = tail call i8* @malloc(i32 trunc (i64 mul (i64 ptrtoint (double* getelementptr (double,
double* null, i32 1) to i64), i64 4) to i32))
  %res70 = bitcast i8* %malloccall69 to [2 x [2 x double]]*
  %tmp71 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 0, i32 0
  store double 0.000000e+00, double* %tmp71
  %tmp72 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new64, i32 0, i32 0, i32 0
  %tmp73 = load double, double* %tmp72
  %tmp74 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 0, i32 0
  %tmp75 = load double, double* %tmp74
  %prod = fmul double %tmp75, %tmp73
  %tmp76 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 0, i32 0
  %tmp77 = load double, double* %tmp76
  %sum = fadd double %prod, %tmp77
  %tmp78 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 0, i32 0
  store double %sum, double* %tmp78
  %tmp79 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new64, i32 0, i32 1, i32 0
  %tmp80 = load double, double* %tmp79
  %tmp81 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 0, i32 1
  %tmp82 = load double, double* %tmp81
  %prod83 = fmul double %tmp82, %tmp80
  %tmp84 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 0, i32 0
  %tmp85 = load double, double* %tmp84
  %sum86 = fadd double %prod83, %tmp85
  %tmp87 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 0, i32 0
  store double %sum86, double* %tmp87
  %tmp88 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new64, i32 0, i32 2, i32 0
  %tmp89 = load double, double* %tmp88
  %tmp90 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 0, i32 2
```

```
%tmp91 = load double, double* %tmp90
%prod92 = fmul double %tmp91, %tmp89
%tmp93 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 0, i32 0
%tmp94 = load double, double* %tmp93
%sum95 = fadd double %prod92, %tmp94
%tmp96 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 0, i32 0
store double %sum95, double* %tmp96
%tmp97 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 0, i32 1
store double 0.000000e+00, double* %tmp97
%tmp98 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new64, i32 0, i32 0, i32 1
%tmp99 = load double, double* %tmp98
%tmp100 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 0, i32 0
%tmp101 = load double, double* %tmp100
%prod102 = fmul double %tmp101, %tmp99
%tmp103 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 0, i32 1
%tmp104 = load double, double* %tmp103
%sum105 = fadd double %prod102, %tmp104
%tmp106 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 0, i32 1
store double %sum105, double* %tmp106
%tmp107 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new64, i32 0, i32 1, i32 1
%tmp108 = load double, double* %tmp107
%tmp109 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 0, i32 1
%tmp110 = load double, double* %tmp109
%prod111 = fmul double %tmp110, %tmp108
%tmp112 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 0, i32 1
%tmp113 = load double, double* %tmp112
%sum114 = fadd double %prod111, %tmp113
%tmp115 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 0, i32 1
store double %sum114, double* %tmp115
%tmp116 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new64, i32 0, i32 2, i32 1
%tmp117 = load double, double* %tmp116
%tmp118 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 0, i32 2
%tmp119 = load double, double* %tmp118
%prod120 = fmul double %tmp119, %tmp117
%tmp121 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 0, i32 1
%tmp122 = load double, double* %tmp121
%sum123 = fadd double %prod120, %tmp122
%tmp124 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 0, i32 1
store double %sum123, double* %tmp124
%tmp125 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 1, i32 0
store double 0.000000e+00, double* %tmp125
%tmp126 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new64, i32 0, i32 0, i32 0
%tmp127 = load double, double* %tmp126
%tmp128 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 1, i32 0
%tmp129 = load double, double* %tmp128
%prod130 = fmul double %tmp129, %tmp127
%tmp131 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 1, i32 0
%tmp132 = load double, double* %tmp131
%sum133 = fadd double %prod130, %tmp132
%tmp134 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 1, i32 0
store double %sum133, double* %tmp134
%tmp135 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new64, i32 0, i32 1, i32 0
%tmp136 = load double, double* %tmp135
%tmp137 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 1, i32 1
%tmp138 = load double, double* %tmp137
%prod139 = fmul double %tmp138, %tmp136
```

```
%tmp140 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 1, i32 0
%tmp141 = load double, double* %tmp140
%sum142 = fadd double %prod139, %tmp141
%tmp143 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 1, i32 0
store double %sum142, double* %tmp143
%tmp144 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new64, i32 0, i32 2, i32 0
%tmp145 = load double, double* %tmp144
%tmp146 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 1, i32 2
%tmp147 = load double, double* %tmp146
%prod148 = fmul double %tmp147, %tmp145
%tmp149 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 1, i32 0
%tmp150 = load double, double* %tmp149
%sum151 = fadd double %prod148, %tmp150
%tmp152 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 1, i32 0
store double %sum151, double* %tmp152
%tmp153 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 1, i32 1
store double 0.000000e+00, double* %tmp153
%tmp154 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new64, i32 0, i32 0, i32 1
%tmp155 = load double, double* %tmp154
%tmp156 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 1, i32 0
%tmp157 = load double, double* %tmp156
%prod158 = fmul double %tmp157, %tmp155
%tmp159 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 1, i32 1
%tmp160 = load double, double* %tmp159
%sum161 = fadd double %prod158, %tmp160
%tmp162 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 1, i32 1
store double %sum161, double* %tmp162
%tmp163 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new64, i32 0, i32 1, i32 1
%tmp164 = load double, double* %tmp163
%tmp165 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 1, i32 1
%tmp166 = load double, double* %tmp165
%prod167 = fmul double %tmp166, %tmp164
%tmp168 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 1, i32 1
%tmp169 = load double, double* %tmp168
%sum170 = fadd double %prod167, %tmp169
%tmp171 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 1, i32 1
store double %sum170, double* %tmp171
%tmp172 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new64, i32 0, i32 2, i32 1
%tmp173 = load double, double* %tmp172
%tmp174 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res45, i32 0, i32 1, i32 2
%tmp175 = load double, double* %tmp174
%prod176 = fmul double %tmp175, %tmp173
%tmp177 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 1, i32 1
%tmp178 = load double, double* %tmp177
%sum179 = fadd double %prod176, %tmp178
%tmp180 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 1, i32 1
store double %sum179, double* %tmp180
%tmp181 = load [2 x [2 x double]], [2 x [2 x double]]* %res70
%tmp182 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %res70, i32 0, i32 0
%indexing183 = load [2 x double], [2 x double]* %tmp182
%inverse = call [2 x [2 x double]]* @inverse([2 x [2 x double]]* %res70, i32 2)
%x_new184 = load [3 x [2 x double]]*, [3 x [2 x double]]** %x_new
%tmp185 = load [3 x [2 x double]], [3 x [2 x double]]* %x_new184
%tmp186 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new184, i32 0, i32 0
%indexing187 = load [2 x double], [2 x double]* %tmp186
%malloccall188 = tail call i8* @malloc(i32 trunc (i64 mul (i64 ptrtoint (double* getelementptr
```

```
(double, double* null, i32 1) to i64), i64 6) to i32))
  %res189 = bitcast i8* %malloccall188 to [2 x [3 x double]]*
  %tmp190 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 0, i32 0
  %tmp191 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new184, i32 0, i32 0, i32 0
  %tmp192 = load double, double* %tmp191
  store double %tmp192, double* %tmp190
  %tmp193 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 0, i32 1
  %tmp194 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new184, i32 0, i32 1, i32 0
  %tmp195 = load double, double* %tmp194
  store double %tmp195, double* %tmp193
  %tmp196 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 0, i32 2
  %tmp197 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new184, i32 0, i32 2, i32 0
  %tmp198 = load double, double* %tmp197
  store double %tmp198, double* %tmp196
  %tmp199 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 1, i32 0
  %tmp200 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new184, i32 0, i32 0, i32 1
  %tmp201 = load double, double* %tmp200
  store double %tmp201, double* %tmp199
  %tmp202 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 1, i32 1
  %tmp203 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new184, i32 0, i32 1, i32 1
  %tmp204 = load double, double* %tmp203
  store double %tmp204, double* %tmp202
  %tmp205 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 1, i32 2
  %tmp206 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new184, i32 0, i32 2, i32 1
  %tmp207 = load double, double* %tmp206
  store double %tmp207, double* %tmp205
  %tmp208 = load [2 x [2 x double]], [2 x [2 x double]]* %inverse
  %tmp209 = load [2 x [3 x double]], [2 x [3 x double]]* %res189
  %tmp210 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 0
  %indexing211 = load [3 x double], [3 x double]* %tmp210
  %malloccall1212 = tail call i8* @malloc(i32 trunc (i64 mul (i64 ptrtoint (double* getelementptr
(double, double* null, i32 1) to i64), i64 6) to i32))
  %res213 = bitcast i8* %malloccall1212 to [2 x [3 x double]]*
  %tmp214 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 0
  store double 0.000000e+00, double* %tmp214
  %tmp215 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 0, i32 0
  %tmp216 = load double, double* %tmp215
  %tmp217 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %inverse, i32 0, i32 0, i32 0
  %tmp218 = load double, double* %tmp217
  %prod219 = fmul double %tmp218, %tmp216
  %tmp220 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 0
  %tmp221 = load double, double* %tmp220
  %sum222 = fadd double %prod219, %tmp221
  %tmp223 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 0
  store double %sum222, double* %tmp223
  %tmp224 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 1, i32 0
  %tmp225 = load double, double* %tmp224
  %tmp226 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %inverse, i32 0, i32 0, i32 1
  %tmp227 = load double, double* %tmp226
  %prod228 = fmul double %tmp227, %tmp225
  %tmp229 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 0
  %tmp230 = load double, double* %tmp229
  %sum231 = fadd double %prod228, %tmp230
  %tmp232 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 0
  store double %sum231, double* %tmp232
  %tmp233 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 1
```

```
store double 0.000000e+00, double* %tmp233
%tmp234 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 0, i32 1
%tmp235 = load double, double* %tmp234
%tmp236 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %inverse, i32 0, i32 0, i32 0
%tmp237 = load double, double* %tmp236
%prod238 = fmul double %tmp237, %tmp235
%tmp239 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 1
%tmp240 = load double, double* %tmp239
%sum241 = fadd double %prod238, %tmp240
%tmp242 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 1
store double %sum241, double* %tmp242
%tmp243 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 1, i32 1
%tmp244 = load double, double* %tmp243
%tmp245 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %inverse, i32 0, i32 0, i32 1
%tmp246 = load double, double* %tmp245
%prod247 = fmul double %tmp246, %tmp244
%tmp248 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 1
%tmp249 = load double, double* %tmp248
%sum250 = fadd double %prod247, %tmp249
%tmp251 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 1
store double %sum250, double* %tmp251
%tmp252 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 2
store double 0.000000e+00, double* %tmp252
%tmp253 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 0, i32 2
%tmp254 = load double, double* %tmp253
%tmp255 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %inverse, i32 0, i32 0, i32 0
%tmp256 = load double, double* %tmp255
%prod257 = fmul double %tmp256, %tmp254
%tmp258 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 2
%tmp259 = load double, double* %tmp258
%sum260 = fadd double %prod257, %tmp259
%tmp261 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 2
store double %sum260, double* %tmp261
%tmp262 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 1, i32 2
%tmp263 = load double, double* %tmp262
%tmp264 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %inverse, i32 0, i32 0, i32 1
%tmp265 = load double, double* %tmp264
%prod266 = fmul double %tmp265, %tmp263
%tmp267 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 2
%tmp268 = load double, double* %tmp267
%sum269 = fadd double %prod266, %tmp268
%tmp270 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 2
store double %sum269, double* %tmp270
%tmp271 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 0
store double 0.000000e+00, double* %tmp271
%tmp272 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 0, i32 0
%tmp273 = load double, double* %tmp272
%tmp274 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %inverse, i32 0, i32 1, i32 0
%tmp275 = load double, double* %tmp274
%prod276 = fmul double %tmp275, %tmp273
%tmp277 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 0
%tmp278 = load double, double* %tmp277
%sum279 = fadd double %prod276, %tmp278
%tmp280 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 0
store double %sum279, double* %tmp280
%tmp281 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 1, i32 0
```

```
%tmp282 = load double, double* %tmp281
%tmp283 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %inverse, i32 0, i32 1, i32 1
%tmp284 = load double, double* %tmp283
%prod285 = fmul double %tmp284, %tmp282
%tmp286 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 0
%tmp287 = load double, double* %tmp286
%sum288 = fadd double %prod285, %tmp287
%tmp289 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 0
store double %sum288, double* %tmp289
%tmp290 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 1
store double 0.000000e+00, double* %tmp290
%tmp291 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 0, i32 1
%tmp292 = load double, double* %tmp291
%tmp293 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %inverse, i32 0, i32 1, i32 0
%tmp294 = load double, double* %tmp293
%prod295 = fmul double %tmp294, %tmp292
%tmp296 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 1
%tmp297 = load double, double* %tmp296
%sum298 = fadd double %prod295, %tmp297
%tmp299 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 1
store double %sum298, double* %tmp299
%tmp300 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 1, i32 1
%tmp301 = load double, double* %tmp300
%tmp302 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %inverse, i32 0, i32 1, i32 1
%tmp303 = load double, double* %tmp302
%prod304 = fmul double %tmp303, %tmp301
%tmp305 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 1
%tmp306 = load double, double* %tmp305
%sum307 = fadd double %prod304, %tmp306
%tmp308 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 1
store double %sum307, double* %tmp308
%tmp309 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 2
store double 0.000000e+00, double* %tmp309
%tmp310 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 0, i32 2
%tmp311 = load double, double* %tmp310
%tmp312 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %inverse, i32 0, i32 1, i32 0
%tmp313 = load double, double* %tmp312
%prod314 = fmul double %tmp313, %tmp311
%tmp315 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 2
%tmp316 = load double, double* %tmp315
%sum317 = fadd double %prod314, %tmp316
%tmp318 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 2
store double %sum317, double* %tmp318
%tmp319 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res189, i32 0, i32 1, i32 2
%tmp320 = load double, double* %tmp319
%tmp321 = getelementptr [2 x [2 x double]], [2 x [2 x double]]* %inverse, i32 0, i32 1, i32 1
%tmp322 = load double, double* %tmp321
%prod323 = fmul double %tmp322, %tmp320
%tmp324 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 2
%tmp325 = load double, double* %tmp324
%sum326 = fadd double %prod323, %tmp325
%tmp327 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 2
store double %sum326, double* %tmp327
%y328 = load [3 x [1 x double]]*, [3 x [1 x double]]** %y
%tmp329 = load [2 x [3 x double]], [2 x [3 x double]]* %res213
%tmp330 = load [3 x [1 x double]], [3 x [1 x double]]* %y328
```

```
  %tmp331 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y328, i32 0, i32 0
  %indexing332 = load [1 x double], [1 x double]* %tmp331
  %malloccall333 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (double* getelementptr
(double, double* null, i32 1) to i64), i64 2) to i32))
  %res334 = bitcast i8* %malloccall333 to [2 x [1 x double]]*
  %tmp335 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %res334, i32 0, i32 0, i32 0
  store double 0.000000e+00, double* %tmp335
  %tmp336 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y328, i32 0, i32 0, i32 0
  %tmp337 = load double, double* %tmp336
  %tmp338 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 0
  %tmp339 = load double, double* %tmp338
  %prod340 = fmul double %tmp339, %tmp337
  %tmp341 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %res334, i32 0, i32 0, i32 0
  %tmp342 = load double, double* %tmp341
  %sum343 = fadd double %prod340, %tmp342
  %tmp344 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %res334, i32 0, i32 0, i32 0
  store double %sum343, double* %tmp344
  %tmp345 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y328, i32 0, i32 1, i32 0
  %tmp346 = load double, double* %tmp345
  %tmp347 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 1
  %tmp348 = load double, double* %tmp347
  %prod349 = fmul double %tmp348, %tmp346
  %tmp350 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %res334, i32 0, i32 0, i32 0
  %tmp351 = load double, double* %tmp350
  %sum352 = fadd double %prod349, %tmp351
  %tmp353 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %res334, i32 0, i32 0, i32 0
  store double %sum352, double* %tmp353
  %tmp354 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y328, i32 0, i32 2, i32 0
  %tmp355 = load double, double* %tmp354
  %tmp356 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 0, i32 2
  %tmp357 = load double, double* %tmp356
  %prod358 = fmul double %tmp357, %tmp355
  %tmp359 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %res334, i32 0, i32 0, i32 0
  %tmp360 = load double, double* %tmp359
  %sum361 = fadd double %prod358, %tmp360
  %tmp362 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %res334, i32 0, i32 0, i32 0
  store double %sum361, double* %tmp362
  %tmp363 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %res334, i32 0, i32 1, i32 0
  store double 0.000000e+00, double* %tmp363
  %tmp364 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y328, i32 0, i32 0, i32 0
  %tmp365 = load double, double* %tmp364
  %tmp366 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 0
  %tmp367 = load double, double* %tmp366
  %prod368 = fmul double %tmp367, %tmp365
  %tmp369 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %res334, i32 0, i32 1, i32 0
  %tmp370 = load double, double* %tmp369
  %sum371 = fadd double %prod368, %tmp370
  %tmp372 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %res334, i32 0, i32 1, i32 0
  store double %sum371, double* %tmp372
  %tmp373 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y328, i32 0, i32 1, i32 0
  %tmp374 = load double, double* %tmp373
  %tmp375 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 1
  %tmp376 = load double, double* %tmp375
  %prod377 = fmul double %tmp376, %tmp374
  %tmp378 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %res334, i32 0, i32 1, i32 0
  %tmp379 = load double, double* %tmp378
```

```
  %sum380 = fadd double %prod377, %tmp379
  %tmp381 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %res334, i32 0, i32 1, i32 0
  store double %sum380, double* %tmp381
  %tmp382 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y328, i32 0, i32 2, i32 0
  %tmp383 = load double, double* %tmp382
  %tmp384 = getelementptr [2 x [3 x double]], [2 x [3 x double]]* %res213, i32 0, i32 1, i32 2
  %tmp385 = load double, double* %tmp384
  %prod386 = fmul double %tmp385, %tmp383
  %tmp387 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %res334, i32 0, i32 1, i32 0
  %tmp388 = load double, double* %tmp387
  %sum389 = fadd double %prod386, %tmp388
  %tmp390 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %res334, i32 0, i32 1, i32 0
  store double %sum389, double* %tmp390
  store [2 x [1 x double]]* %res334, [2 x [1 x double]]** %w
  %x_new391 = load [3 x [2 x double]]*, [3 x [2 x double]]** %x_new
  %w392 = load [2 x [1 x double]]*, [2 x [1 x double]]** %w
  %tmp393 = load [3 x [2 x double]], [3 x [2 x double]]* %x_new391
  %tmp394 = load [2 x [1 x double]], [2 x [1 x double]]* %w392
  %tmp395 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %w392, i32 0, i32 0
  %indexing396 = load [1 x double], [1 x double]* %tmp395
  %malloccall397 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (double* getelementptr
(double, double* null, i32 1) to i64), i64 3) to i32))
  %res398 = bitcast i8* %malloccall397 to [3 x [1 x double]]*
  %tmp399 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res398, i32 0, i32 0, i32 0
  store double 0.000000e+00, double* %tmp399
  %tmp400 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %w392, i32 0, i32 0, i32 0
  %tmp401 = load double, double* %tmp400
  %tmp402 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new391, i32 0, i32 0, i32 0
  %tmp403 = load double, double* %tmp402
  %prod404 = fmul double %tmp403, %tmp401
  %tmp405 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res398, i32 0, i32 0, i32 0
  %tmp406 = load double, double* %tmp405
  %sum407 = fadd double %prod404, %tmp406
  %tmp408 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res398, i32 0, i32 0, i32 0
  store double %sum407, double* %tmp408
  %tmp409 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %w392, i32 0, i32 1, i32 0
  %tmp410 = load double, double* %tmp409
  %tmp411 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new391, i32 0, i32 0, i32 1
  %tmp412 = load double, double* %tmp411
  %prod413 = fmul double %tmp412, %tmp410
  %tmp414 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res398, i32 0, i32 0, i32 0
  %tmp415 = load double, double* %tmp414
  %sum416 = fadd double %prod413, %tmp415
  %tmp417 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res398, i32 0, i32 0, i32 0
  store double %sum416, double* %tmp417
  %tmp418 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res398, i32 0, i32 1, i32 0
  store double 0.000000e+00, double* %tmp418
  %tmp419 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %w392, i32 0, i32 0, i32 0
  %tmp420 = load double, double* %tmp419
  %tmp421 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new391, i32 0, i32 1, i32 0
  %tmp422 = load double, double* %tmp421
  %prod423 = fmul double %tmp422, %tmp420
  %tmp424 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res398, i32 0, i32 1, i32 0
  %tmp425 = load double, double* %tmp424
  %sum426 = fadd double %prod423, %tmp425
  %tmp427 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res398, i32 0, i32 1, i32 0
```

```
  store double %sum426, double* %tmp427
  %tmp428 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %w392, i32 0, i32 1, i32 0
  %tmp429 = load double, double* %tmp428
  %tmp430 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new391, i32 0, i32 1, i32 1
  %tmp431 = load double, double* %tmp430
  %prod432 = fmul double %tmp431, %tmp429
  %tmp433 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res398, i32 0, i32 1, i32 0
  %tmp434 = load double, double* %tmp433
  %sum435 = fadd double %prod432, %tmp434
  %tmp436 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res398, i32 0, i32 1, i32 0
  store double %sum435, double* %tmp436
  %tmp437 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res398, i32 0, i32 2, i32 0
  store double 0.000000e+00, double* %tmp437
  %tmp438 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %w392, i32 0, i32 0, i32 0
  %tmp439 = load double, double* %tmp438
  %tmp440 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new391, i32 0, i32 2, i32 0
  %tmp441 = load double, double* %tmp440
  %prod442 = fmul double %tmp441, %tmp439
  %tmp443 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res398, i32 0, i32 2, i32 0
  %tmp444 = load double, double* %tmp443
  %sum445 = fadd double %prod442, %tmp444
  %tmp446 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res398, i32 0, i32 2, i32 0
  store double %sum445, double* %tmp446
  %tmp447 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %w392, i32 0, i32 1, i32 0
  %tmp448 = load double, double* %tmp447
  %tmp449 = getelementptr [3 x [2 x double]], [3 x [2 x double]]* %x_new391, i32 0, i32 2, i32 1
  %tmp450 = load double, double* %tmp449
  %prod451 = fmul double %tmp450, %tmp448
  %tmp452 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res398, i32 0, i32 2, i32 0
  %tmp453 = load double, double* %tmp452
  %sum454 = fadd double %prod451, %tmp453
  %tmp455 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res398, i32 0, i32 2, i32 0
  store double %sum454, double* %tmp455
  store [3 x [1 x double]]* %res398, [3 x [1 x double]]** %y_lr
  %w456 = load [2 x [1 x double]]*, [2 x [1 x double]]** %w
  %tmp457 = load [2 x [1 x double]], [2 x [1 x double]]* %w456
  %tmp458 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %w456, i32 0, i32 0
  %indexing459 = load [1 x double], [1 x double]* %tmp458
  %tmp460 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %w456, i32 0, i32 0, i32 0
  %indexing461 = load double, double* %tmp460
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.3, i32 0,
i32 0), double %indexing461)
  %printf462 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([2 x i8], [2 x i8]* @fmt.4, i32
0, i32 0))
  %tmp463 = getelementptr [2 x [1 x double]], [2 x [1 x double]]* %w456, i32 0, i32 1, i32 0
  %indexing464 = load double, double* %tmp463
  %printf465 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.3, i32
0, i32 0), double %indexing464)
  %printf466 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([2 x i8], [2 x i8]* @fmt.4, i32
0, i32 0))
  %y_lr467 = load [3 x [1 x double]]*, [3 x [1 x double]]** %y_lr
  %tmp468 = load [3 x [1 x double]], [3 x [1 x double]]* %y_lr467
  %tmp469 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y_lr467, i32 0, i32 0
  %indexing470 = load [1 x double], [1 x double]* %tmp469
  %tmp471 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y_lr467, i32 0, i32 0, i32 0
  %indexing472 = load double, double* %tmp471
```

```
  %printf473 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.3, i32
0, i32 0), double %indexing472)
  %printf474 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([2 x i8], [2 x i8]* @fmt.4, i32
0, i32 0))
  %tmp475 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y_lr467, i32 0, i32 1, i32 0
  %indexing476 = load double, double* %tmp475
  %printf477 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.3, i32
0, i32 0), double %indexing476)
  %printf478 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([2 x i8], [2 x i8]* @fmt.4, i32
0, i32 0))
  %tmp479 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y_lr467, i32 0, i32 2, i32 0
  %indexing480 = load double, double* %tmp479
  %printf481 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.3, i32
0, i32 0), double %indexing480)
  %printf482 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([2 x i8], [2 x i8]* @fmt.4, i32
0, i32 0))
  ret i32 0
}

declare noalias i8* @malloc(i32)

declare [2 x [2 x double]]* @inverse([2 x [2 x double]]*, i32)
```

2. Gradient descent for linear regression source program:

```
func int main() {
  matrix x <3, 1> = [1;2;3];
  matrix y <3, 1> = [4;5;6];
  int n = 3;
  matrix y_current <3, 1>;
  matrix sum_c <1,1>;

  double learning_rate = 0.0001;
  double w_current = 0.0;
  double b_current = 0.0;
  double w_gradient = 0.0;
  double b_gradient = 0.0;

  int i;
  int j;
  double cost;

  for (i = 0; i < 250000; i = i + 1) {
    y_current = (w_current * x) + b_current;
    cost = 0.0;
    for(j = 0; j < n; j = j + 1){
      cost = cost + (y[j,0] - y_current[j,0]) * (y[j,0] - y_current[j,0]);
    }
    cost = cost / n;
    sum_c = sum_col(x .* (y - y_current));
```

```
    w_gradient = -(2.0/n) * sum_c[0,0];
    sum_c = sum_col(y - y_current);
    b_gradient = -(2.0/n) * sum_c[0,0];
    w_current = w_current - (learning_rate * w_gradient);
    b_current = b_current - (learning_rate * b_gradient);
  }
  print(cost);
  print(w_current);
  print(b_current);
  return 0;
}
```

Target program:

```
  ; ModuleID = 'MathLight'
source_filename = "MathLight"

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.1 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt.2 = private unnamed_addr constant [4 x i8] c"%g\0A\00"
@fmt.3 = private unnamed_addr constant [4 x i8] c"%g \00"
@fmt.4 = private unnamed_addr constant [2 x i8] c"\0A\00"

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

declare double @sqrt(double)

declare i32 @abs(i32)

declare double @fabs(double)

declare double @pow(double, double)

declare double @log(double)

define i32 @main() {
entry:
  %x = alloca [3 x [1 x double]]*
  %malloccall = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (double* getelementptr
(double, double* null, i32 1) to i64), i64 3) to i32))
  %res = bitcast i8* %malloccall to [3 x [1 x double]]*
  %tmp = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res, i32 0, i32 0, i32 0
  store double 1.000000e+00, double* %tmp
  %tmp1 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res, i32 0, i32 1, i32 0
  store double 2.000000e+00, double* %tmp1
  %tmp2 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res, i32 0, i32 2, i32 0
  store double 3.000000e+00, double* %tmp2
  store [3 x [1 x double]]* %res, [3 x [1 x double]]** %x
  %y = alloca [3 x [1 x double]]*
  %malloccall3 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (double* getelementptr
(double, double* null, i32 1) to i64), i64 3) to i32))
```

```
  %res4 = bitcast i8* %malloccall3 to [3 x [1 x double]]*
  %tmp5 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res4, i32 0, i32 0, i32 0
  store double 4.000000e+00, double* %tmp5
  %tmp6 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res4, i32 0, i32 1, i32 0
  store double 5.000000e+00, double* %tmp6
  %tmp7 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res4, i32 0, i32 2, i32 0
  store double 6.000000e+00, double* %tmp7
  store [3 x [1 x double]]* %res4, [3 x [1 x double]]** %y
  %n = alloca i32
  store i32 3, i32* %n
  %y_current = alloca [3 x [1 x double]]*
  %sum_c = alloca [1 x [1 x double]]*
  %learning_rate = alloca double
  store double 1.000000e-04, double* %learning_rate
  %w_current = alloca double
  store double 0.000000e+00, double* %w_current
  %b_current = alloca double
  store double 0.000000e+00, double* %b_current
  %w_gradient = alloca double
  store double 0.000000e+00, double* %w_gradient
  %b_gradient = alloca double
  store double 0.000000e+00, double* %b_gradient
  %i = alloca i32
  %j = alloca i32
  %cost = alloca double
  store i32 0, i32* %i
  br label %while

while:                                            ; preds = %merge, %entry
  %i224 = load i32, i32* %i
  %tmp225 = icmp slt i32 %i224, 250000
  br i1 %tmp225, label %while_body, label %merge226

while_body:                                       ; preds = %while
  %w_current8 = load double, double* %w_current
  %x9 = load [3 x [1 x double]]*, [3 x [1 x double]]** %x
  %tmp10 = load [3 x [1 x double]], [3 x [1 x double]]* %x9
  %tmp11 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %x9, i32 0, i32 0
  %indexing = load [1 x double], [1 x double]* %tmp11
  %malloccall12 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (double* getelementptr
(double, double* null, i32 1) to i64), i64 3) to i32))
  %res13 = bitcast i8* %malloccall12 to [3 x [1 x double]]*
  %tmp14 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res13, i32 0, i32 0, i32 0
  %tmp15 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %x9, i32 0, i32 0, i32 0
  %tmp16 = load double, double* %tmp15
  %val = fmul double %w_current8, %tmp16
  store double %val, double* %tmp14
  %tmp17 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res13, i32 0, i32 1, i32 0
  %tmp18 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %x9, i32 0, i32 1, i32 0
  %tmp19 = load double, double* %tmp18
  %val20 = fmul double %w_current8, %tmp19
  store double %val20, double* %tmp17
  %tmp21 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res13, i32 0, i32 2, i32 0
  %tmp22 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %x9, i32 0, i32 2, i32 0
  %tmp23 = load double, double* %tmp22
  %val24 = fmul double %w_current8, %tmp23
```

```
  store double %val24, double* %tmp21
  %b_current25 = load double, double* %b_current
  %tmp26 = load [3 x [1 x double]], [3 x [1 x double]]* %res13
  %tmp27 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res13, i32 0, i32 0
  %indexing28 = load [1 x double], [1 x double]* %tmp27
  %res29 = alloca [3 x [1 x double]]
  %tmp30 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res29, i32 0, i32 0, i32 0
  %tmp31 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res13, i32 0, i32 0, i32 0
  %tmp32 = load double, double* %tmp31
  %val33 = fadd double %b_current25, %tmp32
  store double %val33, double* %tmp30
  %tmp34 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res29, i32 0, i32 1, i32 0
  %tmp35 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res13, i32 0, i32 1, i32 0
  %tmp36 = load double, double* %tmp35
  %val37 = fadd double %b_current25, %tmp36
  store double %val37, double* %tmp34
  %tmp38 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res29, i32 0, i32 2, i32 0
  %tmp39 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res13, i32 0, i32 2, i32 0
  %tmp40 = load double, double* %tmp39
  %val41 = fadd double %b_current25, %tmp40
  store double %val41, double* %tmp38
  store [3 x [1 x double]]* %res29, [3 x [1 x double]]** %y_current
  store double 0.000000e+00, double* %cost
  store i32 0, i32* %j
  br label %while42

while42:                                          ; preds = %while_body43, %while_body
  %j67 = load i32, i32* %j
  %n68 = load i32, i32* %n
  %tmp69 = icmp slt i32 %j67, %n68
  br i1 %tmp69, label %while_body43, label %merge

while_body43:                                     ; preds = %while42
  %cost44 = load double, double* %cost
  %y45 = load [3 x [1 x double]]*, [3 x [1 x double]]** %y
  %j46 = load i32, i32* %j
  %tmp47 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y45, i32 0, i32 %j46, i32 0
  %indexing48 = load double, double* %tmp47
  %y_current49 = load [3 x [1 x double]]*, [3 x [1 x double]]** %y_current
  %j50 = load i32, i32* %j
  %tmp51 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y_current49, i32 0, i32 %j50, i32 0
  %indexing52 = load double, double* %tmp51
  %tmp53 = fsub double %indexing48, %indexing52
  %y54 = load [3 x [1 x double]]*, [3 x [1 x double]]** %y
  %j55 = load i32, i32* %j
  %tmp56 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y54, i32 0, i32 %j55, i32 0
  %indexing57 = load double, double* %tmp56
  %y_current58 = load [3 x [1 x double]]*, [3 x [1 x double]]** %y_current
  %j59 = load i32, i32* %j
  %tmp60 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y_current58, i32 0, i32 %j59, i32 0
  %indexing61 = load double, double* %tmp60
  %tmp62 = fsub double %indexing57, %indexing61
  %tmp63 = fmul double %tmp53, %tmp62
  %tmp64 = fadd double %cost44, %tmp63
  store double %tmp64, double* %cost
  %j65 = load i32, i32* %j
```

```
  %tmp66 = add i32 %j65, 1
  store i32 %tmp66, i32* %j
  br label %while42

merge:                                            ; preds = %while42
  %cost70 = load double, double* %cost
  %n71 = load i32, i32* %n
  %tmp72 = uitofp i32 %n71 to double
  %tmp73 = fdiv double %cost70, %tmp72
  store double %tmp73, double* %cost
  %x74 = load [3 x [1 x double]]*, [3 x [1 x double]]** %x
  %y75 = load [3 x [1 x double]]*, [3 x [1 x double]]** %y
  %y_current76 = load [3 x [1 x double]]*, [3 x [1 x double]]** %y_current
  %tmp77 = load [3 x [1 x double]], [3 x [1 x double]]* %y75
  %tmp78 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y75, i32 0, i32 0
  %indexing79 = load [1 x double], [1 x double]* %tmp78
  %malloccall80 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (double* getelementptr
(double, double* null, i32 1) to i64), i64 3) to i32))
  %res81 = bitcast i8* %malloccall80 to [3 x [1 x double]]*
  %tmp82 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res81, i32 0, i32 0, i32 0
  %tmp83 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y_current76, i32 0, i32 0, i32 0
  %tmp84 = load double, double* %tmp83
  %tmp85 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y75, i32 0, i32 0, i32 0
  %tmp86 = load double, double* %tmp85
  %tmp87 = fsub double %tmp86, %tmp84
  store double %tmp87, double* %tmp82
  %tmp88 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res81, i32 0, i32 1, i32 0
  %tmp89 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y_current76, i32 0, i32 1, i32 0
  %tmp90 = load double, double* %tmp89
  %tmp91 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y75, i32 0, i32 1, i32 0
  %tmp92 = load double, double* %tmp91
  %tmp93 = fsub double %tmp92, %tmp90
  store double %tmp93, double* %tmp88
  %tmp94 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res81, i32 0, i32 2, i32 0
  %tmp95 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y_current76, i32 0, i32 2, i32 0
  %tmp96 = load double, double* %tmp95
  %tmp97 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y75, i32 0, i32 2, i32 0
  %tmp98 = load double, double* %tmp97
  %tmp99 = fsub double %tmp98, %tmp96
  store double %tmp99, double* %tmp94
  %tmp100 = load [3 x [1 x double]], [3 x [1 x double]]* %x74
  %tmp101 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %x74, i32 0, i32 0
  %indexing102 = load [1 x double], [1 x double]* %tmp101
  %malloccall103 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (double* getelementptr
(double, double* null, i32 1) to i64), i64 3) to i32))
  %res104 = bitcast i8* %malloccall103 to [3 x [1 x double]]*
  %tmp105 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res104, i32 0, i32 0, i32 0
  %tmp106 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res81, i32 0, i32 0, i32 0
  %tmp107 = load double, double* %tmp106
  %tmp108 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %x74, i32 0, i32 0, i32 0
  %tmp109 = load double, double* %tmp108
  %tmp110 = fmul double %tmp109, %tmp107
  store double %tmp110, double* %tmp105
  %tmp111 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res104, i32 0, i32 1, i32 0
  %tmp112 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res81, i32 0, i32 1, i32 0
  %tmp113 = load double, double* %tmp112
```

```
  %tmp114 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %x74, i32 0, i32 1, i32 0
  %tmp115 = load double, double* %tmp114
  %tmp116 = fmul double %tmp115, %tmp113
  store double %tmp116, double* %tmp111
  %tmp117 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res104, i32 0, i32 2, i32 0
  %tmp118 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res81, i32 0, i32 2, i32 0
  %tmp119 = load double, double* %tmp118
  %tmp120 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %x74, i32 0, i32 2, i32 0
  %tmp121 = load double, double* %tmp120
  %tmp122 = fmul double %tmp121, %tmp119
  store double %tmp122, double* %tmp117
  %tmp123 = load [3 x [1 x double]], [3 x [1 x double]]* %res104
  %tmp124 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res104, i32 0, i32 0
  %indexing125 = load [1 x double], [1 x double]* %tmp124
  %malloccall1126 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null, i32
1) to i32))
  %res127 = bitcast i8* %malloccall1126 to [1 x [1 x double]]*
  %tmp128 = getelementptr [1 x [1 x double]], [1 x [1 x double]]* %res127, i32 0, i32 0, i32 0
  store double 0.000000e+00, double* %tmp128
  %tmp129 = getelementptr [1 x [1 x double]], [1 x [1 x double]]* %res127, i32 0, i32 0, i32 0
  %tmp130 = getelementptr [1 x [1 x double]], [1 x [1 x double]]* %res127, i32 0, i32 0, i32 0
  %tmp131 = load double, double* %tmp130
  %tmp132 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res104, i32 0, i32 0, i32 0
  %tmp133 = load double, double* %tmp132
  %tmp134 = fadd double %tmp133, %tmp131
  store double %tmp134, double* %tmp129
  %tmp135 = getelementptr [1 x [1 x double]], [1 x [1 x double]]* %res127, i32 0, i32 0, i32 0
  %tmp136 = getelementptr [1 x [1 x double]], [1 x [1 x double]]* %res127, i32 0, i32 0, i32 0
  %tmp137 = load double, double* %tmp136
  %tmp138 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res104, i32 0, i32 1, i32 0
  %tmp139 = load double, double* %tmp138
  %tmp140 = fadd double %tmp139, %tmp137
  store double %tmp140, double* %tmp135
  %tmp141 = getelementptr [1 x [1 x double]], [1 x [1 x double]]* %res127, i32 0, i32 0, i32 0
  %tmp142 = getelementptr [1 x [1 x double]], [1 x [1 x double]]* %res127, i32 0, i32 0, i32 0
  %tmp143 = load double, double* %tmp142
  %tmp144 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res104, i32 0, i32 2, i32 0
  %tmp145 = load double, double* %tmp144
  %tmp146 = fadd double %tmp145, %tmp143
  store double %tmp146, double* %tmp141
  store [1 x [1 x double]]* %res127, [1 x [1 x double]]** %sum_c
  %n147 = load i32, i32* %n
  %tmp148 = uitofp i32 %n147 to double
  %tmp149 = fdiv double 2.000000e+00, %tmp148
  %tmp150 = fsub double -0.000000e+00, %tmp149
  %sum_c151 = load [1 x [1 x double]]*, [1 x [1 x double]]** %sum_c
  %tmp152 = getelementptr [1 x [1 x double]], [1 x [1 x double]]* %sum_c151, i32 0, i32 0, i32 0
  %indexing153 = load double, double* %tmp152
  %tmp154 = fmul double %tmp150, %indexing153
  store double %tmp154, double* %w_gradient
  %y155 = load [3 x [1 x double]]*, [3 x [1 x double]]** %y
  %y_current156 = load [3 x [1 x double]]*, [3 x [1 x double]]** %y_current
  %tmp157 = load [3 x [1 x double]], [3 x [1 x double]]* %y155
  %tmp158 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y155, i32 0, i32 0
  %indexing159 = load [1 x double], [1 x double]* %tmp158
  %malloccall160 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (double* getelementptr
```

```
(double, double* null, i32 1) to i64), i64 3) to i32))
  %res161 = bitcast i8* %malloccall160 to [3 x [1 x double]]*
  %tmp162 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res161, i32 0, i32 0, i32 0
  %tmp163 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y_current156, i32 0, i32 0, i32 0
  %tmp164 = load double, double* %tmp163
  %tmp165 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y155, i32 0, i32 0, i32 0
  %tmp166 = load double, double* %tmp165
  %tmp167 = fsub double %tmp166, %tmp164
  store double %tmp167, double* %tmp162
  %tmp168 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res161, i32 0, i32 1, i32 0
  %tmp169 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y_current156, i32 0, i32 1, i32 0
  %tmp170 = load double, double* %tmp169
  %tmp171 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y155, i32 0, i32 1, i32 0
  %tmp172 = load double, double* %tmp171
  %tmp173 = fsub double %tmp172, %tmp170
  store double %tmp173, double* %tmp168
  %tmp174 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res161, i32 0, i32 2, i32 0
  %tmp175 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y_current156, i32 0, i32 2, i32 0
  %tmp176 = load double, double* %tmp175
  %tmp177 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %y155, i32 0, i32 2, i32 0
  %tmp178 = load double, double* %tmp177
  %tmp179 = fsub double %tmp178, %tmp176
  store double %tmp179, double* %tmp174
  %tmp180 = load [3 x [1 x double]], [3 x [1 x double]]* %res161
  %tmp181 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res161, i32 0, i32 0
  %indexing182 = load [1 x double], [1 x double]* %tmp181
  %malloccall183 = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null, i32
1) to i32))
  %res184 = bitcast i8* %malloccall183 to [1 x [1 x double]]*
  %tmp185 = getelementptr [1 x [1 x double]], [1 x [1 x double]]* %res184, i32 0, i32 0, i32 0
  store double 0.000000e+00, double* %tmp185
  %tmp186 = getelementptr [1 x [1 x double]], [1 x [1 x double]]* %res184, i32 0, i32 0, i32 0
  %tmp187 = getelementptr [1 x [1 x double]], [1 x [1 x double]]* %res184, i32 0, i32 0, i32 0
  %tmp188 = load double, double* %tmp187
  %tmp189 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res161, i32 0, i32 0, i32 0
  %tmp190 = load double, double* %tmp189
  %tmp191 = fadd double %tmp190, %tmp188
  store double %tmp191, double* %tmp186
  %tmp192 = getelementptr [1 x [1 x double]], [1 x [1 x double]]* %res184, i32 0, i32 0, i32 0
  %tmp193 = getelementptr [1 x [1 x double]], [1 x [1 x double]]* %res184, i32 0, i32 0, i32 0
  %tmp194 = load double, double* %tmp193
  %tmp195 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res161, i32 0, i32 1, i32 0
  %tmp196 = load double, double* %tmp195
  %tmp197 = fadd double %tmp196, %tmp194
  store double %tmp197, double* %tmp192
  %tmp198 = getelementptr [1 x [1 x double]], [1 x [1 x double]]* %res184, i32 0, i32 0, i32 0
  %tmp199 = getelementptr [1 x [1 x double]], [1 x [1 x double]]* %res184, i32 0, i32 0, i32 0
  %tmp200 = load double, double* %tmp199
  %tmp201 = getelementptr [3 x [1 x double]], [3 x [1 x double]]* %res161, i32 0, i32 2, i32 0
  %tmp202 = load double, double* %tmp201
  %tmp203 = fadd double %tmp202, %tmp200
  store double %tmp203, double* %tmp198
  store [1 x [1 x double]]* %res184, [1 x [1 x double]]** %sum_c
  %n204 = load i32, i32* %n
  %tmp205 = uitofp i32 %n204 to double
  %tmp206 = fdiv double 2.000000e+00, %tmp205
```

```
  %tmp207 = fsub double -0.000000e+00, %tmp206
  %sum_c208 = load [1 x [1 x double]]*, [1 x [1 x double]]** %sum_c
  %tmp209 = getelementptr [1 x [1 x double]], [1 x [1 x double]]* %sum_c208, i32 0, i32 0, i32 0
  %indexing210 = load double, double* %tmp209
  %tmp211 = fmul double %tmp207, %indexing210
  store double %tmp211, double* %b_gradient
  %w_current212 = load double, double* %w_current
  %learning_rate213 = load double, double* %learning_rate
  %w_gradient214 = load double, double* %w_gradient
  %tmp215 = fmul double %learning_rate213, %w_gradient214
  %tmp216 = fsub double %w_current212, %tmp215
  store double %tmp216, double* %w_current
  %b_current217 = load double, double* %b_current
  %learning_rate218 = load double, double* %learning_rate
  %b_gradient219 = load double, double* %b_gradient
  %tmp220 = fmul double %learning_rate218, %b_gradient219
  %tmp221 = fsub double %b_current217, %tmp220
  store double %tmp221, double* %b_current
  %i222 = load i32, i32* %i
  %tmp223 = add i32 %i222, 1
  store i32 %tmp223, i32* %i
  br label %while

merge226:                                         ; preds = %while
  %cost227 = load double, double* %cost
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.2, i32 0,
i32 0), double %cost227)
  %w_current228 = load double, double* %w_current
  %printf229 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.2, i32
0, i32 0), double %w_current228)
  %b_current230 = load double, double* %b_current
  %printf231 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.2, i32
0, i32 0), double %b_current230)
  ret i32 0
}

declare noalias i8* @malloc(i32)
```

6.2 Test Suites

Our test suites include manual tests and integration tests. Manual tests are specific to scanner and parser. Every time when we modify the parser to adapt to new language design, we use menhir to manually test whether the parser can work correctly without any shift/reduce conflicts or reduce/reduce conflicts.

Other than that, we mainly use integration tests to test whether the entire compiler works. We wrote a program in our language and check whether the output is correct.

6.3 Test Coverage

Everytime we implement a new feature, we add at least one test case which should pass. And we also have failed tests for all the type checking static semantic check supports.

6.4 Test Automation

Adopted from MicroC, we also use a script **'testall.sh'** to automate all the tests, which will be ran every time we run "make" to build the compiler. Also, we can just run the script by itself. The script goes over every tests under the tests folder, where our compiler compiles the program into .ll file firstly. Then LLVM compiler llc translates it into .s file. C compiler compiles the assembly code and the object file to be linked together into an executable. And finally the script runs the executable and compares its output with the desired output.

6.5 Roles

Boya created the test automation infrastructure. Mingye is responsible for the manual tests for parser with menhir. Chunli takes charge of all the failed cases for static semantic checking. Boya and Yuli write all of the passed test cases for all the features.

7. Language Evolution

Our language has changed a lot during the development process. We planned to implement the matrix manipulation, statistic function and file I-O when proposing the language. Then suggested by TA, we decided to focus on matrix and provide rich built-in function because of the time limitation. When designing our language and writing language reference manual, we aimed to provide a built-in function to compute eigenvalue and eigenvector, but we found that eigenvalue computation are very sophisticated and we didn't have a complex number data structure to store the complex eigenvalue. Thus we removed these two built-in functions during the project development process.

Another big change is, when passing a matrix to a function as a formal argument, we used to use matrix with its size. But in that case, the function was not general because the users have to specify the matrix size in formal arguments. We spent tons of time to infer matrix size in semant check and code generation to support passing matrices without sizes as formal arguments. We thought it worked. However, Prof. Edwards pointed out the bug during our presentation that it would cause stack overflow in recursive functions. For this problem, we haven't figured out a good solution for it.

8. Lessons learned

Boya Song: Cooperation is the key to build a good compiler. The front-end and the backend are highly coupled together. To make a feature work, the parser, the semantic checker, and the codegen all needs to work. Every time when we modify a small feature, we debug together to make sure all components work correctly. This is the first time I worked on a project like this and I really enjoyed the cooperation. Also, before I took this course, compilers are always like a black box to me, which is fascinating but mysterious. Since I integrated the whole project and worked on lots of backend stuff, it is clear to me now how a compiler works and how to write a compiler. What's more, don't hesitate to ask for help. Since there's not a lot of resources of LLVM online, it's easy to get stuck when trying to solve challenging problems. Discussing with teammates and TAs are very helpful.

Chunli Fu: The most important thing I learned is how to design a compiler and balance trade-offs with limited time and resources. When you try to figure out a problem, others may come up. When it's not possible to solve one problem in a perfect way, how to make decisions is very important, which needs overall and detailed understanding of the project. Besides, at the beginning of the class, I considered Ocaml as a complex language which is not easy to understand. However, after using it to finish the whole project, I realize this language is natural for writing a compiler. I've written an interpreter using C++ before, which is very complicated with a lot of troubles not related to the intrinsic thoughts of language design. Using Ocaml to write compiler can really save your time!

Mingye Chen: Think deeper when you designing syntax and implementing the front-end of a language. I wrote parser and part of the scanner and felt I learned a lot from fixing conflicts, taking care of corner cases, improving the efficiency and making codes more concise of the front-end. Parser is an important component in the whole pipeline, small changes may produce larger effects since it may modify the AST. So deeper understanding of the whole pipeline is encouraged. Also, Ocaml will be a good friend for writing front-end of a language after you figure out how things work.

Yuli Han: The most important thing I learned in this project is how to work using a language with limited tutorial and resource. I wrote part of the code generation code in this project and found Ocaml coding is very tricky. I always met trouble in figuring out the cause for bugs. The online resource for llvm API is also very limited. I have to discuss with teammates and emailed TA very often to make sure we have used the API correctly. Besides this I also learned a lot of

other things like the basic structure of a compiler, how to write test, communication with teammates. Last but not least, start early! Most of our work were done in the last week so we didn't have much time to go to office hour.

9. Appendix

8.1 Scanner.mll Author: Mingye Chen and Yuli Han

```
{ open Parser }

rule token = parse
  [' ' '\t' '\r' '\n'] { token lexbuf }
       (* comment *)
       | "/*" { long_comment lexbuf }
       | "//" { short_comment lexbuf }
       (* operators *)
       | '+' { PLUS }
       | '-' { MINUS }
       | '*' { TIMES }
       | '/' { DIVIDE }
       | '=' { ASSIGN }
       | '^' { POWER }
       | '|' { ABS }
       | ''' { TRANSPOSE }
       | ".*" { DOTTIMES }
       | "./" { DOTDIVIDE }
       | ".^" { DOTPOWER }
       (* logical operators *)
       | '>' { GT }
       | '<' { LT }
       | "<=" { LSEQ }
       | ">=" { GTEQ }
       | "==" { EQ }
       | "!=" { NEQ }
       | "&&" { AND }
       | "||" { OR }
       | '!' { NOT }
       (* special operators *)
       | '[' { LSQ }
       | ']' { RSQ }
       | '(' { LPA }
       | ')' { RPA }
       | '{' { LBR }
       | '}' { RBR }
       | ';' { SEMI }
       | ',' { COMMA }
       | ':' { COLON }
```

```
        (* data types *)
        | "int" { INT }
        | "double" { DOUBLE }
        | "boolean" { BOOLEAN }
        | "string" { STRING }
        | "matrix" { MATRIX }
        | "void" { VOID }
        (* control flow *)
        | "if" { IF }
        | "else" { ELSE }
        | "for" { FOR }
        | "while" { WHILE }
        | "continue" { CONTINUE }
        | "break" { BREAK }
        | "func" { FUNC }
        | "return" { RETURN }
        (* constants *)
        | "PI" { PI }
        | "True" { TRUE }
        | "False" { FALSE }
        (* literals *)
        | ['0'-'9']+ as lit { INTLIT(int_of_string lit) }
        | '"' ([^ '"']* as lit) '"' { STRLIT(lit) }
        | (['0'-'9']*)'.'(['0'-'9']+) as lit { DOUBLELIT(float_of_string lit) }
        | (['0'-'9']+)'.'(['0'-'9']*) as lit { DOUBLELIT(float_of_string lit) }
        | ['a'-'z']['a'-'z' '0'-'9' 'A'-'Z' '_']* as lit { ID(lit) }
        (* special cases *)
        | eof { EOF }
        | _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }

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

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

8.2 parser.mly Author: Mingye Chen

```
%{
open Ast
%}

%token PLUS MINUS TIMES DIVIDE POWER ASSIGN ABS TRANSPOSE DOTTIMES DOTDIVIDE DOTPOWER
%token GT LT LSEQ GTEQ EQ NEQ AND OR NOT
%token LSQ RSQ LPA RPA LBR RBR SEMI COMMA COLON
```

```
%token INT DOUBLE BOOLEAN STRING MATRIX VOID
%token IF ELSE FOR WHILE CONTINUE BREAK FUNC RETURN
%token PI TRUE FALSE
%token EOF

%token<int> INTLIT
%token<float> DOUBLELIT
%token<string> STRLIT ID

%nonassoc NOELSE
%nonassoc ELSE
%nonassoc COLON
%right ASSIGN
%left OR
%left AND
%left EQ NEQ
%left LT GT LSEQ GTEQ
%left PLUS MINUS
%left TIMES DIVIDE DOTTIMES DOTDIVIDE
%right NOT NEG
%left TRANSPOSE POWER DOTPOWER

%start program
%type <Ast.program> program

%%

program:
  decls EOF { $1 }

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

fdecl:
   FUNC data_type ID LPA formals_opt RPA LBR vdecl_list stmt_list RBR
     {
       { data_type = $2;
        function_name = $3;
        arguments = $5;
        local_vars = List.rev $8;
        body = List.rev $9 }
     }

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

formal_list:
```

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

data_type:
    INT { Int }
  | BOOLEAN { Bool }
  | VOID { Void }
  | MATRIX { Matrix }
  | STRING { String }
  | DOUBLE { Double }

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

/* In this way, data_type will be uniform (benefit the whole pipeline) and resolve
shift/reduce conflict */
vdecl:
    data_type ID SEMI { check_size_normal_return_bind $1 $2 Noexpr }
  | data_type ID matrix_size SEMI { check_size_matrix_return_bind $1 $2 $3 Noexpr }
  | data_type ID ASSIGN expr SEMI { check_size_normal_return_bind $1 $2 $4 }
  | data_type ID matrix_size ASSIGN expr SEMI { check_size_matrix_return_bind $1 $2 $3 $5 }

matrix_size:
    LT INTLIT GT { (1, $2) }
  | LT INTLIT COMMA INTLIT GT { ($2, $4) }

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

statement:
    expr SEMI { Expr $1 }
  | BREAK SEMI { Break Noexpr }
  | CONTINUE SEMI { Continue Noexpr }
  | RETURN SEMI { Return Noexpr }
  | RETURN expr SEMI { Return $2 }
  | LBR stmt_list RBR { Block(List.rev $2) }
  | IF LPA expr RPA statement %prec NOELSE { If($3, $5, Block([])) }
  | IF LPA expr RPA statement ELSE statement    { If($3, $5, $7) }
  | FOR LPA expr_opt SEMI expr SEMI expr_opt RPA statement
     { For($3, $5, $7, $9) }
  | FOR LPA INT ID ASSIGN expr RPA statement
     { ForRange($4, $6, $8) }
  | WHILE LPA expr RPA statement { While($3, $5) }

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

```
expr:
    INTLIT           { IntLit($1) }
  | DOUBLELIT        { DoubleLit($1) }
  | STRLIT           { StrLit($1) }
  | TRUE             { BoolLit(true) }
  | FALSE            { BoolLit(false) }
  | ID               { Id($1) }
  | PI               { DoubleLit(3.1415926535) }
  | 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 POWER  expr { Binop($1, Pow,   $3) }
  | expr DOTTIMES  expr { Binop($1, Dotmul,  $3) }
  | expr DOTDIVIDE expr { Binop($1, Dotdiv,  $3) }
  | expr DOTPOWER  expr { Binop($1, Dotpow,  $3) }
  | expr EQ     expr { Binop($1, Equal, $3) }
  | expr NEQ    expr { Binop($1, Neq,   $3) }
  | expr LT     expr { Binop($1, Less,  $3) }
  | expr LSEQ   expr { Binop($1, Leq,   $3) }
  | expr GT     expr { Binop($1, Greater, $3) }
  | expr GTEQ   expr { Binop($1, Geq,   $3) }
  | expr AND    expr { Binop($1, And,   $3) }
  | expr OR     expr { Binop($1, Or,    $3) }
  | expr COLON  expr { Range($1, $3) }
  | LSQ ID COMMA ID RSQ           { MatrixOp($2, Comma, $4) }
  | LSQ ID SEMI ID RSQ            { MatrixOp($2, Semi,  $4) }
  | LSQ matrix_lit RSQ            { MatrixLit($2) }
  | ID LSQ expr RSQ               { Matrix1DElement($1, $3) }
  | ID LSQ expr COMMA expr RSQ    { Matrix2DElement($1, $3, $5) }
  | MINUS expr %prec NEG { Unop(Neg, $2) }
  | NOT expr         { Unop(Not, $2) }
  | ABS expr ABS     { Unop(Abs, $2) }
  | expr TRANSPOSE   { Unop(Transpose, $1) }
  | ID ASSIGN expr   { Assign($1, $3) }
  | ID LSQ expr COMMA expr RSQ ASSIGN expr   { Matrix2DModify($1, ($3,$5), $8) }
  | ID LSQ expr RSQ ASSIGN expr              { Matrix1DModify($1, $3, $6) }
  | ID LPA actuals_opt RPA { Call($1, $3) }
  | LPA expr RPA { $2 }

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

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

matrix_lit:
    matrix_lit_1D                  { [|$1|] }
```

```
  | matrix_lit_2D                  { $1 }

matrix_primitive:
    DOUBLELIT                      { $1 }
  | INTLIT                         { float_of_int $1 }
  | MINUS DOUBLELIT                { -. $2 }
  | MINUS INTLIT                   { float_of_int (-$2) }

matrix_lit_1D:
    matrix_primitive { [|$1|] }
  | matrix_lit_1D COMMA matrix_primitive { Array.append $1 [|$3|] }

matrix_lit_2D:
  | matrix_lit_1D SEMI matrix_lit_1D { [|$1; $3|]}
  | matrix_lit_2D SEMI matrix_lit_1D {Array.append $1 [|$3|]}
```

8.3 ast.ml Author: Mingye Chen

```
type operator = Add | Sub | Mult | Div | Pow
              | Dotmul | Dotdiv | Dotpow
              | Greater | Geq | Leq | Less
              | Neq | Equal
              | And | Or
              | Comma | Semi

type unary_operator = Not | Neg | Abs | Transpose

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

type expr =
    Binop of expr * operator * expr
  | Unop of unary_operator * expr
  | IntLit of int
  | DoubleLit of float
  | StrLit of string
  | BoolLit of bool
  | Range of expr * expr
  | MatrixLit of float array array
  | MatrixOp of string * operator * string
  | Matrix1DElement of string * expr
  | Matrix2DElement of string * expr * expr
  | Matrix1DModify of string * expr * expr
  | Matrix2DModify of string * (expr * expr) * expr
  | Id of string
  | Assign of string * expr
  | Call of string * expr list
  | Noexpr
```

```
type bind = data_type * string * (int * int) * expr

type statement =
         Block of statement list
  | Expr of expr
  | If of expr * statement * statement
  | For of expr * expr * expr * statement
  | ForRange of string * expr * statement
  | While of expr * statement
  | Continue of expr
  | Break of expr
  | Return of expr

type function_declare = {
  data_type : data_type;
       function_name : string;
       arguments : bind list;
  local_vars : bind list;
       body : statement list;
}

type program = bind list * function_declare list

(* Parser helper functions *)

let check_size_matrix_return_bind data_type variable_name matrix_size expr =
  match data_type with
    Matrix -> (data_type, variable_name, matrix_size, expr)
    | _ -> failwith("only matrix type can declare its size")

let check_size_normal_return_bind data_type variable_name expr =
  match data_type with
    Matrix -> failwith("should assign the size for matrix type")
    | _ -> (data_type, variable_name, (-1, -1), expr)

(* Pretty-printing functions *)

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

```
  | Or -> "||"
  | Comma -> ","
  | Semi -> ":"
  | Pow -> "^"
  | Dotmul -> ".*"
  | Dotdiv -> "./"
  | Dotpow -> ".^"

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

let rec string_of_expr = function
    IntLit(l) -> string_of_int l
  | DoubleLit(l) -> string_of_float l
  | BoolLit(true) -> "true"
  | BoolLit(false) -> "false"
  | StrLit(s) -> s
  | Id(s) -> s
  | Binop(e1, o, e2) ->
      string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
  | Unop(o, e) -> string_of_uop o ^ string_of_expr e
  | Assign(v, e) -> v ^ " = " ^ string_of_expr e
  | Call(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | Noexpr -> ""
  | MatrixLit(_) -> "MatrixLit"
  | Matrix1DElement(s, e) -> "Matrix1DElement " ^ s ^ "[" ^ string_of_expr(e) ^ "]"
  | Matrix2DElement(s, e1, e2) ->
      "SMatrix2DElement " ^ s ^ "[" ^ string_of_expr(e1) ^ ", " ^ string_of_expr(e2) ^ "]"
  | Range(e1, e2) -> "Range: " ^ string_of_expr(e1) ^ ": " ^ string_of_expr(e2)
  | MatrixOp(m1, o, m2) -> "MatrixOp " ^ m1 ^ string_of_op(o) ^ m2
  | Matrix1DModify(s, e1, e2) ->
      s ^ "[" ^ string_of_expr(e1) ^ "]" ^ " = " ^ string_of_expr(e2)
  | Matrix2DModify(s, (e1,e2), e3) ->
      s ^ "[" ^ string_of_expr(e1) ^ ", " ^ string_of_expr(e2) ^ "]" ^ " = " ^
string_of_expr(e3)

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
  | _ -> "other stmt"

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

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

let string_of_fdecl fdecl =
  let sndOfQuadruple = fun (_, y, _, _) -> y in
  string_of_typ fdecl.data_type ^ " " ^
  fdecl.function_name ^ "(" ^ String.concat ", " (List.map sndOfQuadruple fdecl.arguments) ^
  ")\n{\n" ^
  String.concat "" (List.map string_of_vdecl fdecl.local_vars) ^
  String.concat "" (List.map string_of_stmt fdecl.body) ^
  "}\n"

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

let string_of_size (s) =
  let (r, c) = s in
 "(" ^ string_of_int(r) ^ ", " ^ string_of_int(c) ^ ")"
```

8.4 sast.ml Author: Chunli Fu

```
open Ast

type sexpr = (data_type * (int * int)) * sx
and sx =
    SIntLit of int
  | SDoubleLit of float
  | SBoolLit of bool
  | SStrLit of string
  | SRange of sexpr * sexpr
  | SMatrixLit of float array array
  | SMatrix1DElement of string * sexpr
  | SMatrix2DElement of string * sexpr * sexpr
  | SMatrix1DModify of string * sexpr * sexpr
  | SMatrix2DModify of string * (sexpr * sexpr) * sexpr
  | SId of string
```

```
  | SBinop of sexpr * operator * sexpr
  | SUnop of unary_operator * sexpr
  | SMatrixOp of string * operator * string
  | SAssign of string * sexpr
  | SCall of string * sexpr list
  | SNoexpr

type sbind = data_type * string * (int * int) * sexpr

type sstmt =
    SBlock of sstmt list
  | SExpr of sexpr
  | SReturn of sexpr
  | SIf of sexpr * sstmt * sstmt
  | SFor of sexpr * sexpr * sexpr * sstmt
  | SForRange of string * sexpr * sstmt
  | SWhile of sexpr * sstmt
  | SContinue of sexpr
  | SBreak of sexpr

type sfunc_decl = {
    styp : data_type;
    ssize : (int * int);
    sfname : string;
    sargs : sbind list;
    slocals : sbind list;
    sbody : sstmt list;
  }

type sprogram = sbind list * sfunc_decl list

(* Pretty-printing functions *)

let rec string_of_sexpr ((t, (r, c)), e) =
  "(" ^ string_of_typ t ^ " with size (" ^ string_of_int r ^ ", " ^ string_of_int c ^ ") : "
^
  (match e with
    SIntLit(l) -> string_of_int l
  | SBoolLit(true) -> "true"
  | SBoolLit(false) -> "false"
  | SStrLit(s) -> s
  | SDoubleLit(l) -> string_of_float(l)
  | SId(s) -> s
  | SBinop(e1, o, e2) ->
      string_of_sexpr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_sexpr e2
  | SUnop(o, e) -> string_of_uop o ^ string_of_sexpr e
  | SAssign(v, e) -> v ^ " = " ^ string_of_sexpr e
  | SCall(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_sexpr el) ^ ")"
  | SNoexpr -> ""
```

```
  | SMatrixLit(_) -> "SMatrixLit"
  | SMatrix1DElement(s, e) -> "SMatrix1DElement " ^ s ^ "[" ^ string_of_sexpr(e) ^ "]"
  | SMatrix2DElement(s, e1, e2) ->
      "SMatrix2DElement " ^ s ^ "[" ^ string_of_sexpr(e1) ^ ", " ^ string_of_sexpr(e2) ^ "]"
  | SMatrix1DModify(s, e1, e2) ->
      s ^ "[" ^ string_of_sexpr(e1) ^ "]" ^ " = " ^ string_of_sexpr(e2)
  | SMatrix2DModify(s, (e1,e2), e3) ->
      s ^ "[" ^ string_of_sexpr(e1) ^ ", " ^ string_of_sexpr(e2) ^ "]" ^ " = " ^
string_of_sexpr(e3)
  | SRange(e1, e2) -> "SRange: " ^ string_of_sexpr(e1) ^ ": " ^ string_of_sexpr(e2)
  | SMatrixOp(m1, o, m2) -> "SMatrixOp " ^ m1 ^ string_of_op(o) ^ m2
         ) ^ ")"

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
  | SBreak(e) -> "break " ^ string_of_sexpr e
  | SContinue(e) -> "continue " ^ string_of_sexpr e
  | SForRange(_,_,_) -> "SForRange"

let string_of_svdecl (t, id, _, sexpr) = string_of_typ t ^ " " ^ id ^ " " ^ string_of_sexpr
sexpr ^";\n"

let string_of_sfdecl fdecl =
  let sndOfTriple = fun (_, y, _, _) -> y in
  string_of_typ fdecl.styp ^ " " ^
  fdecl.sfname ^ "(" ^ String.concat ", " (List.map sndOfTriple fdecl.sargs) ^
  ")\n{\n" ^
  String.concat "" (List.map string_of_svdecl fdecl.slocals) ^
  String.concat "" (List.map string_of_sstmt fdecl.sbody) ^
  "}\n"

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

8.5 semant.ml Author: Chunli Fu

```
open Ast
```

```
open Sast

module StringMap = Map.Make(String)

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

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

  check_binds "global" globals;

  (* Collect function declarations for built-in functions: no bodies *)
  let built_in_decls =
    let add_bind map (data_ty, name, args) = StringMap.add name {
      data_type = data_ty;
      function_name = name;
      arguments = args;
      local_vars = []; body = [] } map
    in List.fold_left add_bind StringMap.empty [ (Void, "print", [(String, "x", (-1, -1),
Noexpr)]);
                                                 (Double, "sqrt", [(Double, "x", (-1, -1),
Noexpr)]);
                                                 (Double, "log", [(Double, "x", (-1, -1),
Noexpr)]);
                                                 (Matrix, "fill", [(Int, "r", (-1, -1),
Noexpr); (Int, "c", (-1, -1), Noexpr); (Double, "num", (-1, -1), Noexpr);]);
                                                 (Matrix, "inv", [(Matrix, "x", (-1, -1),
Noexpr)]);
                                                 (Double, "det", [(Matrix, "x", (-1, -1),
Noexpr)]);
                                                 (Double, "tr", [(Matrix, "x", (-1, -1),
Noexpr)]);
                                                 (Double, "max_eigvalue", [(Matrix, "x",
(-1, -1), Noexpr)]);
                                                 (Double, "norm2", [(Matrix, "x", (-1, -1),
Noexpr)]);
                                                 (Int, "sizeof_row", [(Matrix, "x", (-1,
-1), Noexpr)]);
                                                 (Int, "sizeof_col", [(Matrix, "x", (-1,
```

```
-1), Noexpr)]);
                                                (Double, "norm1", [(Matrix, "x", (-1, -1),
Noexpr)]);
                                                (Matrix, "sum_row", [(Matrix, "x", (-1,
-1), Noexpr)]);
                                                (Matrix, "sum_col", [(Matrix, "x", (-1,
-1), Noexpr)]);
                                                (Matrix, "mean_row", [(Matrix, "x", (-1,
-1), Noexpr)]);
                                                (Matrix, "mean_col", [(Matrix, "x", (-1,
-1), Noexpr)])];
  in

  (* Add function name to symbol table *)
  let add_func map fd =
    let built_in_err = "function " ^ fd.function_name ^ " may not be defined"
    and dup_err = "duplicate function " ^ fd.function_name
    and make_err er = raise (Failure er)
    and n = fd.function_name (* 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 is_number typ =
    if typ = Int || typ = Double || typ = Bool then true
    else false
  in
  let higher_type t1 t2 =
    if t1 = Double || t2 = Double then Double
    else if t1 = Int || t2 = Int then Int
    else Bool
  in
  (* Raise an exception if the given rvalue type cannot be assigned to
       the given lvalue type *)
  let check_assign lvaluet rvaluet err overload =
    let (t1,s1) = lvaluet and (t2,s2) = rvaluet in
    let (r1,c1) = s1 and (r2,c2) = s2 in
```

```
    if t1 = t2 && t1 != Matrix then rvaluet
    else if t1 = t2 && t1 = Matrix && s1=s2 then lvaluet
    else if t1 = t2 && t1 = Matrix && ((r1 = r2 || r1 = -1 || r2 = -1) && (c1 = c2 || c1 =
-1 || c2 = -1)) then (t1,((max r1 r2),(max c2 c2)))
    else if (is_number t1) = true && (is_number t2) = true then ((higher_type t1 t2), s2)
    else if overload = true then rvaluet
    else raise (Failure err)
  in

  let type_of_identifier symbols s =
    try StringMap.find s symbols
    with Not_found -> raise (Failure ("undeclared identifier " ^ s))
  in

  let get_num_from_expr e =
    match e with
      SIntLit l -> l
    | _ -> raise (Failure("IntLit expeceted"))
  in
  let get_builtin_size fname args =
    match fname with
      "size" -> (true, (1, 2))
    | "inv" | "det" | "tr" -> let (( _, s), _) = List.hd args in (true, s)
    | "sum_col" | "mean_col" -> let ((_, (_, c)), _) = List.hd args in (true, (1, c))
    | "sum_row" | "mean_row" -> let ((_, (r, _)), _) = List.hd args in (true, (r, 1))
    | "fill" -> let (_, e1) = List.hd args and (_, e2) = List.hd (List.tl args) in (true,
(get_num_from_expr e1, get_num_from_expr e2))
    | _ -> (false, (-1, -1))
  in
  (* Return a semantically-checked expression, i.e., with a type *)
  let rec expr symbols = function
      IntLit  l -> ((Int, (-1, -1)), SIntLit l)
    | DoubleLit l -> ((Double, (-1, -1)), SDoubleLit l)
    | BoolLit l  -> ((Bool, (-1, -1)), SBoolLit l)
    | StrLit  l -> ((String, (-1, -1)), SStrLit l)
    | MatrixLit l ->
        let row_size = Array.length(l) in
        let col_size = Array.length(l.(0)) in
        ((Matrix, (row_size, col_size)), SMatrixLit l)
    | Noexpr     -> ((Void, (-1, -1)), SNoexpr)
    | Id s       -> (type_of_identifier symbols s, SId s)
    | Matrix1DElement(m, i) as ex ->
        let (t, s) = type_of_identifier symbols m in
        let ((t2, s2), e2') = expr symbols i in
        if t2 != Int then
          raise (Failure ("illegal 1D matrix index type " ^ string_of_typ(t2) ^
                          "for matrix " ^ m))
        else if t != Matrix then
          raise (Failure ("illegal 1D matrix operation, matrix type expected, get "
                          ^ string_of_typ(t)))
```

```
        else
          let (_, c) = s in
          let ty = match e2' with
          SIntLit l when c != -1 && l >= c -> raise (Failure ("expression " ^ string_of_expr
ex ^
                                                              " out of boundary, matrix size: (" ^
string_of_int(c) ^ ")"))
          | _ -> (Double, (-1, -1))
          in (ty, SMatrix1DElement(m, ((t2, s2), e2')))
    | Matrix2DElement(m, i1, i2) as ex ->
        let (t, s) = type_of_identifier symbols m in
        let ((t1, s1), e1') = expr symbols i1 in
        let ((t2, s2), e2') = expr symbols i2 in
        if t1 != Int || t2 != Int then
          raise (Failure ("illegal 2D matrix index type [" ^ string_of_typ(t1) ^ ", " ^
                          string_of_typ(t2) ^ "] for matrix " ^ m))
        else if t != Matrix then
          raise (Failure ("illegal 2D matrix operation, matrix type expected, get " ^
string_of_typ(t)))
        else
          let (r, c) = s in
          let _ = match e1' with
          SIntLit l when r != -1 && l >= r -> raise (Failure ("expression " ^ string_of_expr
ex ^
                                                              " out of boundary, matrix size: (" ^
string_of_int(r) ^ ", " ^ string_of_int(c)))
          | _ -> ""
          and _ = match e2' with
          SIntLit l when c != -1 && l >= c -> raise (Failure ("expression " ^ string_of_expr
ex ^
                                                              " out of boundary, matrix size: (" ^
string_of_int(r) ^ ", " ^ string_of_int(c)))
          | _ -> ""
          in ((Double, (-1, -1)), SMatrix2DElement(m, ((t1, s1), e1'), ((t2, s2), e2')))
    | Matrix1DModify(m, i, e) as ex ->
        let (t, s) = type_of_identifier symbols m in
        let ((t1, s1), e1') = expr symbols i in
        let ((t2, s2), e2') = expr symbols e in
        if t1 != Int then
          raise (Failure ("illegal 1D matrix index type " ^ string_of_typ(t1) ^
                          "for matrix " ^ m))
        else if t != Matrix then
          raise (Failure ("illegal 1D matrix operation, matrix type expected, get "
                           ^ string_of_typ(t)))
        else if is_number(t2) = false then
          raise (Failure ("illegal 1D matrix assignment, number expected, get " ^
string_of_typ(t2) ))
        else
          let (_, c) = s in
          let ty = match e1' with
```

```
          SIntLit l when c != -1 && l >= c -> raise (Failure ("expression " ^ string_of_expr
ex ^
                                                             " out of boundary, matrix size: ("
^  string_of_int(c) ^ ")"))
          | _ -> (Double, (-1, -1))
          in (ty, SMatrix1DModify(m, ((t1, s1), e1'), ((t2, s2), e2')))
    | Matrix2DModify(m, (i1, i2), e) as ex ->
        let (t, s) = type_of_identifier symbols m in
        let ((t1, s1), e1') = expr symbols i1 in
        let ((t2, s2), e2') = expr symbols i2 in
        let ((t3, s3), e3') = expr symbols e in
        if t1 != Int || t2 != Int then
          raise (Failure ("illegal 2D matrix index type [" ^ string_of_typ(t1) ^ ", " ^
                          string_of_typ(t2) ^ "] for matrix " ^ m))
        else if t != Matrix then
          raise (Failure ("illegal 2D matrix operation, matrix type expected, get " ^
string_of_typ(t)))
        else if is_number(t3) = false then
          raise (Failure ("illegal 2D matrix assignment, number expected, get " ^
string_of_typ(t3) ))
        else
          let (r, c) = s in
          let _ = match e1' with
          SIntLit l when r != -1 && l >= r -> raise (Failure ("expression " ^ string_of_expr
ex ^
                                                             " out of boundary, matrix size: (" ^
string_of_int(r) ^ ", " ^ string_of_int(c)))
          | _ -> ""
          and _ = match e2' with
          SIntLit l when c != -1 && l >= c -> raise (Failure ("expression " ^ string_of_expr
ex ^
                                                             " out of boundary, matrix size: (" ^
string_of_int(r) ^ ", " ^ string_of_int(c)))
          | _ -> ""
          in ((Double, (-1, -1)), SMatrix2DModify(m, (((t1, s1), e1'), ((t2, s2), e2')),
((t3, s3), e3')))
    | Range(e1, e2) as ex ->   (*TODO*)
          let ((t1, s1), e1') = expr symbols e1
          and ((t2, s2), e2') = expr symbols e2 in
          if t1 != Int || t2 != Int then
            raise (Failure ("illegal Range type, int : int expected, get "
                            ^ string_of_typ(t1) ^ " : " ^ string_of_typ(t2) ^ " in " ^
                            string_of_expr(e1) ^ " : " ^ string_of_expr(e2)))
          else
            let a1 = match e1' with
            SIntLit l -> l
            | _ -> 0
            and a2 = match e2' with
            SIntLit l -> l
            | _ -> 1
```

```
            in
            if a1 >= a2 then raise( Failure("Invalid argument in " ^ string_of_expr(ex)))
            else ((Int, (-1, -1)), SRange(((t1, s1), e1'), ((t2, s2), e2')))
    | Assign(var, e) as ex ->
        let (lt,s1) = type_of_identifier symbols var
        and ((rt,s2), e') = expr symbols e in
        let err = "illegal assignment " ^ string_of_typ lt ^ string_of_size(s1) ^ " = " ^
          string_of_typ rt^ string_of_size(s2) ^ " in " ^ string_of_expr ex in
        let check_res = check_assign (lt,s1) (rt,s2) err false in
        let _ = StringMap.add var check_res symbols
        in (check_res, SAssign(var, ((rt,s2), e')))
    | Unop(op, e) as ex ->
        let ((t, s), e') = expr symbols e in
        let (r, c) = s in
        let ty = match op with
          Neg when t = Int || t = Double -> (t, s)
        | Not when t = Bool -> (Bool, s)
        | Abs when t = Int || t = Double || t = Matrix -> (t, s)
        | Transpose when t = Matrix -> (t, (c, r))
        | _ -> raise (Failure ("illegal unary operator: " ^
                               string_of_uop op ^ string_of_typ t ^
                               " in " ^ string_of_expr ex))
        in (ty, SUnop(op, ((t,s), e')))
    | Binop(e1, op, e2) as e ->
        let ((t1, s1), e1') = expr symbols e1
        and ((t2, s2), e2') = expr symbols e2 in
        let (r1, c1) = s1 and (r2, c2) = s2 in
        (* All binary operators require operands of the same type *)
        let same = t1 = t2 in
        let sameMsize = t1 = Matrix && t2 = Matrix && (s1 = s2 || s1 = (-1, -1) || s2 = (-1,
-1)) in
        let sameRTsize = if s1 = (-1, -1) || s2 = (-1, -1) then (-1, -1) else s1 in
        let matchMsize = t1 = Matrix && t2 = Matrix && (c1 = r2 || s1 = (-1, -1) || s2 =
(-1, -1)) in
        let matchRTsize = if s1 = (-1, -1) || s2 = (-1, -1) then (-1, -1) else (r1, c2) in
        (* Determine expression type based on operator and operand types *)
        let ty = match op with
          Add | Sub | Mult | Div            when is_number t1 && is_number t2 ->
((higher_type t1 t2), s1)
        | Add | Sub                       when same && t1 = Matrix && sameMsize -> (Matrix,
sameRTsize)
        | Mult                           when same && t1 = Matrix && matchMsize -> (Matrix,
matchRTsize)
        | Add | Sub | Mult        when (t1 = Double || t1 = Int) && t2 = Matrix -> (Matrix,
s2)
        | Add | Sub | Mult | Div  when t1 = Matrix && (t2 = Double || t2 = Int) -> (Matrix,
s1)
        | Pow                         when (t1 = Int || t1 = Double) && (t2 = Int || t2 =
Double) -> (higher_type t1 t2, s1)
        | Dotmul | Dotdiv | Dotpow when t1 = Matrix && t2 = Matrix && sameMsize -> (Matrix,
```

```
sameRTsize)
        | Equal | Neq            when same                      -> (Bool, s1)
        | Less | Leq | Greater | Geq
                  when same && (t1 = Int || t1 = Double)        -> (Bool, s1)
        | And | Or when same && t1 = Bool -> (Bool, s1)
        | _ -> raise (
                 Failure ("illegal binary operator: " ^
                   string_of_typ t1 ^ " of size (" ^ string_of_int(r1) ^ ","
^string_of_int(c1) ^ ") " ^ string_of_op op ^ " " ^
                   string_of_typ t2 ^ " of size (" ^ string_of_int(r2) ^ ","
^string_of_int(c2) ^ ") in " ^ string_of_expr e))
        in (ty, SBinop(((t1,s1), e1'), op, ((t2,s2), e2')))
    | MatrixOp(e1, op, e2) as e ->
        let (t1, (r1, c1)) = type_of_identifier symbols e1
        and (t2, (r2, c2)) = type_of_identifier symbols e2 in
        let getnum n1 n2 = if n1 = -1 || n2 = -1 then -1 else n1 in
        let getsum n1 n2 = if n1 = -1 || n2 = -1 then -1 else n1 + n2 in
        let ty = match op with
          Comma when t1 = Matrix && t2 = Matrix && (r1 = r2 || r1 = -1 || r2 = -1) ->
(Matrix, ((getnum r1 r2), (getsum c1 c2)))
        | Semi  when t1 = Matrix && t2 = Matrix && (c1 = c2 || c1 = -1 || c2 = -1) ->
(Matrix, ((getsum r1 r2), (getnum c1 c2)))
        | _ -> raise( Failure("illegal Matrix Concat operator: " ^
                        string_of_typ t1 ^ " of size (" ^ string_of_int(r1) ^ ","
^string_of_int(c1) ^ ") " ^ string_of_op op ^ " " ^
                        string_of_typ t2 ^ " of size (" ^ string_of_int(r2) ^ ","
^string_of_int(c2) ^ ") in " ^ string_of_expr e))
        in (ty, SMatrixOp(e1, op, e2))
    (* | Call(fname, args) as call ->  *)
    | Call(fname, args) as call ->
        let fd = find_func fname in
        let param_length = List.length fd.arguments in
        if List.length args != param_length then
          (* raise (Failure ("expecting " ^ string_of_int param_length )) *)
          raise (Failure ("expecting " ^ string_of_int param_length ^
                          " arguments in " ^ string_of_expr call))
        else let check_call (ft, _, (row_size, col_size), _) e =
          let ((et,es), e') = expr symbols e in
          let err = "illegal argument found " ^ string_of_typ et ^ " " ^ string_of_expr e ^
            " expected " ^ string_of_typ ft ^ " in " ^ string_of_expr call in
          let overload = (fname = "print")
          in (check_assign (ft,(row_size,col_size)) (et,es) err overload, e')
        in
        let args' = List.map2 check_call fd.arguments args in
        let (is_builtin, builtin_size) = get_builtin_size fname args' in
        let ret_size =
        if is_builtin then builtin_size
        else
          if fd.data_type = Matrix then
            let get_new_args formal arg =
```

```
              let ((t, size), _) = arg in
              let (_, name, _, e) = formal in
              (t, name, size, e)
            in
            let get_new_f func args =
              {
                data_type = func.data_type;
                function_name = func.function_name;
                arguments = List.map2 get_new_args func.arguments args;
                local_vars = func.local_vars;
                body = func.body;
              } in
            let new_f = (get_new_f fd args') in
            let sf = (check_function true new_f) in
            sf.ssize
          else (-1, -1)
        in
        ((fd.data_type, ret_size), SCall(fname, args'))
  and
  check_bool_expr symbols e =
    let ((t', s), e') = expr symbols e
    and err = "expected Boolean expression in " ^ string_of_expr e
    in if t' != Bool then raise (Failure err) else ((t', s), e')
  and
  (* Return a semantically-checked statement i.e. containing sexprs *)
  get_assign_expr symbols b =
    let (typ, name, size, e) = b in
    let e' =
    if e = Noexpr then ((Void, (-1, -1)), SNoexpr)
    else let (_, e_assign) = expr symbols (Assign (name, e)) in
    match e_assign with
      SAssign(_, e') -> e'
      | _ -> raise (Failure ("internal error: declare and assign"))
    in
    (typ, name, size, e')
  and
  (**** Check functions ****)
  check_function infer_type func =
    (* Make sure no formals or locals are void or duplicates *)
    check_binds "arguments" func.arguments;
    check_binds "local_vars" func.local_vars;

    (* Build local symbol table of variables for this function *)
    let symbols = List.fold_left (fun m (ty, name, (row_size, col_size), _) -> StringMap.add
name (ty, (row_size, col_size)) m)
                        StringMap.empty (globals @ func.arguments @ func.local_vars )
    in
    let rec check_stmt symbols = function
      Expr e -> SExpr (expr symbols e)
    | If(p, b1, b2) -> SIf(check_bool_expr symbols p, check_stmt symbols b1, check_stmt
```

```
symbols b2)
    | For(e1, e2, e3, st) ->
      SFor(expr symbols e1, check_bool_expr symbols e2, expr symbols e3, check_stmt symbols
st)
    | ForRange(var, e, st) ->     (*TODO*)
      SForRange(var, expr symbols e, check_stmt symbols st)
    | While(p, s) -> SWhile(check_bool_expr symbols p, check_stmt symbols s)
    | Return e ->
        let ((t, s), e') = expr symbols e in
        if t = func.data_type then SReturn ((t, s), e')
        else raise (
            Failure ("return gives " ^ string_of_typ t ^ " expected " ^
            string_of_typ 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 symbols = function
            [Return _ as s] -> [check_stmt symbols s]
          | Return _ :: _   -> raise (Failure "nothing may follow a return")
          | Block sl :: ss  -> check_stmt_list symbols (sl @ ss) (* Flatten blocks *)
          | s :: ss         -> check_stmt symbols s :: check_stmt_list symbols ss
          | []              -> []
        in SBlock(check_stmt_list symbols sl)
    | Break e ->
        let ((t, s), e') = expr symbols e in
        if e' != SNoexpr then raise (Failure("break stmt should include Noexpr, " ^
                                             string_of_expr e ^ " found"))
        else SBreak((t, s), e')
    | Continue e ->
        let ((t, s), e') = expr symbols e in
        if e' != SNoexpr then raise (Failure("continue stmt should include Noexpr, " ^
                                             string_of_expr e ^ " found"))
        else SContinue((t, s), e')
    in
    (* Return a variable from our local symbol table *)
    { styp = func.data_type;
      sfname = func.function_name;
      sargs = List.map (get_assign_expr symbols) func.arguments;
      slocals  = List.map (get_assign_expr symbols) func.local_vars;
      sbody =
      if infer_type = false then
        let no_return = (List.length func.body) = 0 ||
          match (List.hd (List.rev func.body)) with
            Return(_) -> false
            | _ -> true
        in
        if func.data_type != Void && no_return then
          raise( Failure("function " ^ func.function_name ^ " has no return"))
        else
```

```
        match check_stmt symbols (Block func.body) with
          SBlock(sl) -> sl
          | _ -> raise (Failure ("internal error: block didn't become a block?"))
      else [];
      ssize =
      if infer_type = true && func.data_type = Matrix then
        let sblock =
          match check_stmt symbols (Block func.body) with
            SBlock(sl) -> sl
            | _ -> raise (Failure ("internal error: block didn't become a block?"))
        in
        let last_stmt = (List.hd (List.rev sblock)) in
        match last_stmt with
          SReturn((_, s), _) -> s
          | _ -> raise (Failure ("last stmt in func is not return in func " ^
func.function_name))
      else (-1, -1);
    }
  in
  let global_symbols = List.fold_left (fun m (ty, name, (row_size, col_size), _) ->
    StringMap.add name (ty, (row_size, col_size)) m) StringMap.empty (globals)
  in (List.map (get_assign_expr global_symbols) globals, List.map (check_function false)
functions)
```

8.6 codegen.ml Author: Boya Song and Yuli Han

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

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

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

Detailed documentation on the OCaml LLVM library:

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

*)

module L = Llvm
module A = Ast
open Sast

module StringMap = Map.Make(String)

(* translate : Sast.program -> Llvm.module *)
let translate (globals, functions) =
```

```
let context    = L.global_context () in

(* Create the LLVM compilation module into which
   we will generate code *)
let the_module = L.create_module context "MathLight" 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 array_t m n = (L.array_type (L.array_type (L.double_type context) n) m) in

(* Return the LLVM type for a MicroC 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 -> L.pointer_type (array_t 10 10)
in

let global_vars = Hashtbl.create 12 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 sqrt_t : L.lltype =
    L.function_type double_t [| double_t |] in
let sqrt_func : L.llvalue =
    L.declare_function "sqrt" sqrt_t the_module in

let abs_t : L.lltype =
    L.function_type i32_t [| i32_t |] in
let abs_func : L.llvalue =
    L.declare_function "abs" abs_t the_module in

let fabs_t : L.lltype =
    L.function_type double_t [| double_t |] in
let fabs_func : L.llvalue =
    L.declare_function "fabs" fabs_t the_module in

let pow_t : L.lltype =
    L.function_type double_t [| double_t; double_t |] in
let pow_func : L.llvalue =
    L.declare_function "pow" pow_t the_module in
```

```
  let log_t : L.lltype =
      L.function_type double_t [| double_t|] in
  let log_func : L.llvalue =
      L.declare_function "log" log_t the_module in
  let matrix_size = Hashtbl.create 12 in

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

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

    let int_format_str = L.build_global_stringptr "%d\n" "fmt" builder
    and string_format_str = L.build_global_stringptr "%s\n" "fmt" builder
    and double_format_str = L.build_global_stringptr "%g\n" "fmt" builder
    and matrix_format_str = L.build_global_stringptr "%g " "fmt" builder
    and return_format_str = L.build_global_stringptr "\n" "fmt" builder in
    (* Construct the function's "locals": formal arguments and locally
       declared variables.  Allocate each on the stack, initialize their
       value, if appropriate, and remember their values in the "locals" map *)

    (* Return the value for a variable or formal argument.
       Check local names first, then global names *)
    let local_vars = Hashtbl.create 10 in
    let lookup n = try Hashtbl.find local_vars n
                   with Not_found -> Hashtbl.find global_vars n
    in

    let rec range i j = if i > j then [] else i :: (range (i+1) j)

    in

    let extract_element ptr i j builder = (L.build_load (L.build_gep ptr [|L.const_int i32_t
0; L.const_int i32_t i; L.const_int i32_t j|] "tmp" builder) "tmp" builder) in

    let get_matrix_row ptr builder = L.array_length (L.type_of (L.build_load ptr "tmp"
builder)) in
    let get_matrix_col ptr builder = L.array_length (L.type_of (L.build_load (L.build_gep
ptr [|L.const_int i32_t 0; L.const_int i32_t 0|] "tmp" builder) "indexing"  builder)) in
```

```
    let get_element_address ptr i j builder =  (L.build_gep ptr [|L.const_int i32_t 0;
L.const_int i32_t i; L.const_int i32_t j|]  "tmp" builder) in
    (* Construct code for an expression; return its value *)
    let rec expr builder (((_, (g_row, g_col)), e) : sexpr) = match e with
        SStrLit i  -> let m = String.length i in
                      let str = L.build_alloca (L.array_type i8_t (m + 1)) "res" builder
in
                      let idxs = range 0 (m - 1) in
                        List.iter (fun idx -> ignore(L.build_store  (L.const_int i8_t
(int_of_char (String.get i idx))) (L.build_gep str [|L.const_int i32_t 0; L.const_int i32_t
idx|]  "tmp" builder)  builder)) idxs; ignore(L.build_store  (L.const_int i8_t 0)
(L.build_gep str [|L.const_int i32_t 0; L.const_int i32_t m|]  "tmp" builder)  builder); str
      | SIntLit i -> L.const_int i32_t i
      | SBoolLit i -> L.const_int i1_t (if i then 1 else 0)
      | SDoubleLit i -> L.const_float double_t i
      | SMatrixLit s -> let m = Array.length s and n = Array.length (Array.get s 0) in
                      let matrix = L.build_malloc (array_t m n) "res" builder in
                         let row_idxs = range 0 (m - 1) in
                        let col_idxs = range 0 (n-1) in
                        List.iter (fun row_idx -> List.iter (fun idx ->
ignore(L.build_store  (L.const_float double_t (Array.get (Array.get s row_idx) idx))
(L.build_gep matrix [|L.const_int i32_t 0; L.const_int i32_t row_idx; L.const_int i32_t
idx|]  "tmp" builder)  builder)) col_idxs; ) row_idxs; matrix
      | SMatrixOp (s1, op, s2) -> let s1' = L.build_load (lookup s1) s1 builder and s2' =
L.build_load (lookup s2) s2 builder in
                                  let m1 = get_matrix_row s1' builder and
                                  n1 = get_matrix_col s1' builder and
                                  m2 = get_matrix_row s2' builder and
                                  n2 = get_matrix_col s2' builder in
                                  let n = n1 + n2 and m = m1 + m2 in
                                  (match op with
                                  A.Comma -> let result = L.build_malloc (array_t m1 n)
"res" builder in
                                    for i = 0 to m1-1 do
                                      for j = 0 to n1+n2-1 do
                                        if j < n1 then
                                          ignore(L.build_store (extract_element s1' i j
builder) (get_element_address result i j builder)  builder)
                                        else
                                          ignore(L.build_store (extract_element s2' i (j-n1)
builder) (get_element_address result i j builder)  builder)
                                      done
                                    done;
                                    result
                                  | _ -> let result = L.build_malloc (array_t m n1) "res"
builder in
                                    for i = 0 to m1 + m2 - 1 do
                                      for j = 0 to n1 do
                                        if i < m1 then
                                          ignore(L.build_store  (extract_element s1' i j
```

```
builder) (get_element_address result i j builder)  builder)
                                        else
                                          ignore(L.build_store  (extract_element s2' (i-m1)
j builder) (get_element_address result i j builder)  builder)
                                      done
                                    done;
                                    result)
      | SBinop (e1, op, e2) ->
        let e1' = expr builder e1 and
        e2' = expr builder e2 in
        let type_of_e1 = L.classify_type (L.type_of e1') and
        type_of_e2 = L.classify_type (L.type_of e2') in
        if type_of_e1 = L.TypeKind.Double && type_of_e2 = L.TypeKind.Double then
        (match op with
          A.Add     -> L.build_fadd e1' e2' "tmp" builder
        | A.Sub     -> L.build_fsub e1' e2' "tmp" builder
        | A.Mult    -> L.build_fmul e1' e2' "tmp" builder
        | A.Div     -> L.build_fdiv e1' e2' "tmp" builder
        | A.Equal   -> L.build_fcmp L.Fcmp.Oeq e1' e2' "tmp" builder
        | A.Neq     -> L.build_fcmp L.Fcmp.One e1' e2' "tmp" builder
        | A.Less    -> L.build_fcmp L.Fcmp.Olt e1' e2' "tmp" builder
        | A.Leq     -> L.build_fcmp L.Fcmp.Ole e1' e2' "tmp" builder
        | A.Greater -> L.build_fcmp L.Fcmp.Ogt e1' e2' "tmp" builder
        | A.Geq     -> L.build_fcmp L.Fcmp.Oge e1' e2' "tmp" builder
        | A.Pow     -> L.build_call pow_func [| e1'; e2' |] "pow" builder
        | _ -> raise (Failure "internal error: semant should have rejected on doubles")
        )  else if type_of_e1 = L.TypeKind.Double && type_of_e2 = L.TypeKind.Integer then
        (match op with
          A.Add     -> L.build_fadd e1' (L.build_uitofp e2' double_t "tmp" builder) "tmp"
builder
        | A.Sub     -> L.build_fsub e1' (L.build_uitofp e2' double_t "tmp" builder) "tmp"
builder
        | A.Mult    -> L.build_fmul e1' (L.build_uitofp e2' double_t "tmp" builder) "tmp"
builder
        | A.Div     -> L.build_fdiv e1' (L.build_uitofp e2' double_t "tmp" builder) "tmp"
builder
        | A.Equal   -> L.build_fcmp L.Fcmp.Oeq e1' (L.build_uitofp e2' double_t "tmp"
builder) "tmp" builder
        | A.Neq     -> L.build_fcmp L.Fcmp.One e1' (L.build_uitofp e2' double_t "tmp"
builder) "tmp" builder
        | A.Less    -> L.build_fcmp L.Fcmp.Olt e1' (L.build_uitofp e2' double_t "tmp"
builder) "tmp" builder
        | A.Leq     -> L.build_fcmp L.Fcmp.Ole e1' (L.build_uitofp e2' double_t "tmp"
builder) "tmp" builder
        | A.Greater -> L.build_fcmp L.Fcmp.Ogt e1' (L.build_uitofp e2' double_t "tmp"
builder) "tmp" builder
        | A.Geq     -> L.build_fcmp L.Fcmp.Oge e1' (L.build_uitofp e2' double_t "tmp"
builder) "tmp" builder
        | A.Pow     -> L.build_call pow_func [| e1'; (L.build_uitofp e2' double_t "tmp"
builder) |] "pow" builder
```

```
        | _ -> raise (Failure "internal error: semant should have rejected")
        ) else if type_of_e1 = L.TypeKind.Integer && type_of_e2 = L.TypeKind.Double then
        (match op with
          A.Add     -> L.build_fadd (L.build_uitofp e1' double_t "tmp" builder) e2' "tmp"
builder
        | A.Sub     -> L.build_fsub (L.build_uitofp e1' double_t "tmp" builder) e2' "tmp"
builder
        | A.Mult    -> L.build_fmul (L.build_uitofp e1' double_t "tmp" builder) e2' "tmp"
builder
        | A.Div     -> L.build_fdiv (L.build_uitofp e1' double_t "tmp" builder) e2' "tmp"
builder
        | A.Equal   -> L.build_fcmp L.Fcmp.Oeq (L.build_uitofp e1' double_t "tmp" builder)
e2' "tmp" builder
        | A.Neq     -> L.build_fcmp L.Fcmp.One (L.build_uitofp e1' double_t "tmp" builder)
e2' "tmp" builder
        | A.Less    -> L.build_fcmp L.Fcmp.Olt (L.build_uitofp e1' double_t "tmp" builder)
e2' "tmp" builder
        | A.Leq     -> L.build_fcmp L.Fcmp.Ole (L.build_uitofp e1' double_t "tmp" builder)
e2' "tmp" builder
        | A.Greater -> L.build_fcmp L.Fcmp.Ogt (L.build_uitofp e1' double_t "tmp" builder)
e2' "tmp" builder
        | A.Geq     -> L.build_fcmp L.Fcmp.Oge (L.build_uitofp e1' double_t "tmp" builder)
e2' "tmp" builder
        | A.Pow     -> L.build_call pow_func [| (L.build_uitofp e1' double_t "tmp" builder);
e2' |] "pow" builder
        | _ -> raise (Failure "internal error: semant should have rejected")
        ) else if type_of_e1 = L.TypeKind.Integer && type_of_e2 = L.TypeKind.Integer then
        (match op with
          A.Add     -> L.build_add e1' e2' "tmp" builder
        | A.Sub     -> L.build_sub e1' e2' "tmp" builder
        | A.Mult    -> L.build_mul e1' e2' "tmp" builder
        | A.Div     -> L.build_sdiv e1' e2' "tmp" builder
        | A.Equal   -> L.build_icmp L.Icmp.Eq e1' e2' "tmp" builder
        | A.Neq     -> L.build_icmp L.Icmp.Ne e1' e2' "tmp" builder
        | A.Less    -> L.build_icmp L.Icmp.Slt e1' e2' "tmp" builder
        | A.Leq     -> L.build_icmp L.Icmp.Sle e1' e2' "tmp" builder
        | A.Greater -> L.build_icmp L.Icmp.Sgt e1' e2' "tmp" builder
        | A.Geq     -> L.build_icmp L.Icmp.Sge e1' e2' "tmp" builder
        | A.Pow     -> L.build_fptosi (L.build_call pow_func [| (L.build_uitofp e1' double_t
"tmp" builder); (L.build_uitofp e2' double_t "tmp" builder) |] "pow" builder) i32_t "tmp"
builder
        | A.And     -> L.build_and e1' e2' "tmp" builder
        | A.Or      -> L.build_or e1' e2' "tmp" builder
        | _ -> raise (Failure "internal error: semant should have rejected")
        ) else if (type_of_e1 = L.TypeKind.Double || type_of_e1 = L.TypeKind.Integer) &&
type_of_e2 = L.TypeKind.Pointer then
        (match op with
          A.Mult    ->   let m = (get_matrix_row e2' builder) and n = (get_matrix_col e2'
builder) in
                         let result = L.build_malloc (array_t m n) "res" builder  in
```

```
                              for i = 0 to (m - 1) do
                                for j = 0 to (n - 1) do
                                  if type_of_e1 = L.TypeKind.Double then
                                  ignore(L.build_store (L.build_fmul e1' (extract_element
e2' i j builder) "val" builder)  (get_element_address result i j builder)  builder)
                                  else ignore(L.build_store (L.build_fmul (L.build_uitofp
e1' double_t "tmp" builder) (extract_element e2' i j builder) "val" builder)
(get_element_address result i j builder)  builder);
                                done
                              done;
                          result
          | A.Add   ->    let m = (get_matrix_row e2' builder) and n = (get_matrix_col e2'
builder) in
                          let result = L.build_alloca (array_t m n) "res" builder  in
                              for i = 0 to (m - 1) do
                                for j = 0 to (n - 1) do
                                  if type_of_e1 = L.TypeKind.Double then
                                  ignore(L.build_store (L.build_fadd e1' (extract_element
e2' i j builder) "val" builder)  (get_element_address result i j builder)  builder)
                                  else ignore(L.build_store (L.build_fadd (L.build_uitofp
e1' double_t "tmp" builder) (extract_element e2' i j builder) "val" builder)
(get_element_address result i j builder)  builder);
                                done
                              done;
                          result
          | A.Sub   ->    let m = (get_matrix_row e2' builder) and n = (get_matrix_col e2'
builder) in
                          let result = L.build_alloca (array_t m n) "res" builder  in
                              for i = 0 to (m - 1) do
                                for j = 0 to (n - 1) do
                                  if type_of_e1 = L.TypeKind.Double then
                                  ignore(L.build_store (L.build_fsub e1' (extract_element
e2' i j builder) "val" builder)  (get_element_address result i j builder)  builder)
                                  else ignore(L.build_store (L.build_fsub (L.build_uitofp
e1' double_t "tmp" builder) (extract_element e2' i j builder) "val" builder)
(get_element_address result i j builder)  builder);
                                done
                              done;
                          result
          | A.Div   ->    let m = (get_matrix_row e2' builder) and n = (get_matrix_col e2'
builder) in
                          let result = L.build_alloca (array_t m n) "res" builder  in
                              for i = 0 to (m - 1) do
                                for j = 0 to (n - 1) do
                                  if type_of_e1 = L.TypeKind.Double then
                                  ignore(L.build_store (L.build_fdiv e1' (extract_element
e2' i j builder) "val" builder)  (get_element_address result i j builder)  builder)
                                  else ignore(L.build_store (L.build_fdiv (L.build_uitofp
e1' double_t "tmp" builder) (extract_element e2' i j builder) "val" builder)
(get_element_address result i j builder)  builder);
```

```
                                done
                            done;
                        result
        | _         ->    raise (Failure "internal error: semant does not support operation
on given type.")
       ) else if type_of_e1 = L.TypeKind.Pointer && (type_of_e2 = L.TypeKind.Double ||
type_of_e2 = L.TypeKind.Integer) then (
         match op with
         A.Mult -> let m = (get_matrix_row e1' builder) and n = (get_matrix_col e1'
builder) in
                        let result = L.build_malloc (array_t m n) "res" builder  in
                            for i = 0 to (m - 1) do
                              for j = 0 to (n - 1) do
                                if type_of_e2 = L.TypeKind.Double then
                                ignore(L.build_store (L.build_fmul e2' (extract_element
e1' i j builder) "val" builder)  (get_element_address result i j builder)  builder)
                                else ignore(L.build_store (L.build_fmul (L.build_uitofp e2'
double_t "tmp" builder) (extract_element e1' i j builder) "val" builder)
(get_element_address result i j builder)  builder);
                                done
                            done;
                        result
         | A.Add -> let m = (get_matrix_row e1' builder) and n = (get_matrix_col e1'
builder) in
                        let result = L.build_alloca (array_t m n) "res" builder  in
                            for i = 0 to (m - 1) do
                              for j = 0 to (n - 1) do
                                if type_of_e2 = L.TypeKind.Double then
                                ignore(L.build_store (L.build_fadd e2' (extract_element
e1' i j builder) "val" builder)  (get_element_address result i j builder)  builder)
                                else ignore(L.build_store (L.build_fadd (L.build_uitofp e2'
double_t "tmp" builder) (extract_element e1' i j builder) "val" builder)
(get_element_address result i j builder)  builder);
                                done
                            done;
                        result
         | A.Sub -> let m = (get_matrix_row e1' builder) and n = (get_matrix_col e1'
builder) in
                        let result = L.build_alloca (array_t m n) "res" builder  in
                            for i = 0 to (m - 1) do
                              for j = 0 to (n - 1) do
                                if type_of_e2 = L.TypeKind.Double then
                                ignore(L.build_store (L.build_fsub (extract_element e1' i
j builder) e2' "val" builder)  (get_element_address result i j builder)  builder)
                                else ignore(L.build_store (L.build_fsub (extract_element
e1' i j builder) (L.build_uitofp e2' double_t "tmp" builder) "val" builder)
(get_element_address result i j builder)  builder);
                                done
                            done;
                        result
```

```
          | A.Div -> let m = (get_matrix_row e1' builder) and n = (get_matrix_col e1'
builder) in
                          let result = L.build_alloca (array_t m n) "res" builder  in
                              for i = 0 to (m - 1) do
                                for j = 0 to (n - 1) do
                                  if type_of_e2 = L.TypeKind.Double then
                                  ignore(L.build_store (L.build_fdiv (extract_element e1' i
j builder) e2' "val" builder)  (get_element_address result i j builder)  builder)
                                   else ignore(L.build_store (L.build_fdiv (extract_element
e1' i j builder) (L.build_uitofp e2' double_t "tmp" builder) "val" builder)
(get_element_address result i j builder)  builder);
                                  done
                              done;
                          result
          | _         ->    raise (Failure "internal error: semant does not support operation
on given type.")
        )else if type_of_e1 = L.TypeKind.Pointer && type_of_e2 = L.TypeKind.Pointer then
        (match op with
          A.Add     ->    let m = (get_matrix_row e1' builder) and n = (get_matrix_col e1'
builder) in
                          let result = L.build_malloc (array_t m n) "res" builder  in
                              for i = 0 to m - 1 do
                                for j = 0 to (n - 1) do
                                  ignore(L.build_store (L.build_fadd (extract_element e1' i
j builder) (extract_element e2' i j builder) "tmp" builder) (get_element_address result i j
builder)  builder);
                                  done
                              done;
                          result
        | A.Sub   ->      let m = (get_matrix_row e1' builder) and n = (get_matrix_col e1'
builder) in
                          let result = L.build_malloc (array_t m n) "res" builder  in
                              for i = 0 to m - 1 do
                                for j = 0 to (n - 1) do
                                  ignore(L.build_store (L.build_fsub (extract_element e1' i
j builder) (extract_element e2' i j builder) "tmp" builder) (get_element_address result i j
builder)  builder);
                                  done
                              done;
                          result
        | A.Mult  ->      let matrix1 = L.build_load e1' "tmp" builder and matrix2 =
L.build_load e2' "tmp" builder in
                          let m = L.array_length (L.type_of matrix1) and k = L.array_length
(L.type_of matrix2) and
                          n = L.array_length (L.type_of (L.build_load (L.build_gep e2'
[|L.const_int i32_t 0; L.const_int i32_t 0|] "tmp" builder) "indexing"  builder)) in
                            let result = L.build_malloc (array_t m n) "res" builder  in
                            for i = 0 to m - 1 do
                              for j = 0 to (n - 1) do
                                ignore(L.build_store (L.const_float double_t 0.0)
```

```
(get_element_address result i j builder) builder);
                                for p = 0 to k - 1 do
                                  let prod = L.build_fmul (extract_element e1' i p builder)
(extract_element e2' p j builder) "prod" builder in
                                  let sum = L.build_fadd prod (extract_element result i j
builder) "sum" builder in
                                  ignore(L.build_store sum (get_element_address result i j
builder) builder);
                                done
                              done
                            done;
                        result
        | A.Dotmul  ->    let m = (get_matrix_row e1' builder) and n = (get_matrix_col e1'
builder) in
                          let result = L.build_malloc (array_t m n) "res" builder  in
                              for i = 0 to m - 1 do
                                for j = 0 to (n - 1) do
                                  ignore(L.build_store (L.build_fmul (extract_element e1' i
j builder) (extract_element e2' i j builder) "tmp" builder) (get_element_address result i j
builder)  builder);
                                done
                              done;
                        result
        | A.Dotdiv  ->    let m = (get_matrix_row e1' builder) and n = (get_matrix_col e1'
builder) in
                          let result = L.build_malloc (array_t m n) "res" builder  in
                              for i = 0 to m - 1 do
                                for j = 0 to (n - 1) do
                                  ignore(L.build_store (L.build_fdiv (extract_element e1' i
j builder) (extract_element e2' i j builder) "tmp" builder) (get_element_address result i j
builder)  builder);
                                done
                              done;
                        result
        | _ -> raise (Failure "internal error: semant should have rejected")
        ) else raise (Failure "internal error: semant does not support operation on given
type.")
      | SId s       -> L.build_load (lookup s) s builder
      | SAssign (s, (((typ, _), _) as e)) -> let e' = expr builder e in
                           (match typ with
                           A.String -> let m = get_matrix_row e' builder in
                            let dest = L.build_bitcast (lookup s) (L.pointer_type
(L.pointer_type (L.array_type i8_t m))) "tmp" builder in
                             ignore(L.build_store e' dest builder); e'
                           | _ -> ignore(L.build_store e' (lookup s) builder); e')
      | SNoexpr     -> L.const_int i32_t 0
      | SMatrix1DElement (name, e) -> let matrix_ptr = (L.build_load (lookup name) name
builder) and e' = expr builder e in
                    (L.build_load (L.build_gep matrix_ptr [|L.const_int i32_t 0; L.const_int
i32_t 0; e'|] "tmp" builder) "indexing"  builder)
```

```
      | SMatrix2DElement (name, e1, e2) -> let matrix_ptr = (L.build_load (lookup name) name
builder) and e1' = expr builder e1 and e2' = expr builder e2 in
                    (L.build_load (L.build_gep matrix_ptr [|L.const_int i32_t 0; e1'; e2'|]
"tmp" builder) "indexing" builder)
      | SMatrix1DModify (name, e1, e2) -> let matrix_ptr = (L.build_load (lookup name) name
builder) and e1' = expr builder e1 and                                  e2' = expr builder
e2 in
                                      (L.build_store e2' (L.build_gep matrix_ptr
[|L.const_int i32_t 0; e1'|] "tmp" builder) builder)
      | SMatrix2DModify (name, (e_row, e_col), e) -> let matrix_ptr = (L.build_load (lookup
name) name builder) and er' = expr builder e_row and ec' = expr builder e_col and e' = expr
builder e in
                    (L.build_store e' (L.build_gep matrix_ptr [|L.const_int i32_t 0; er';
ec'|] "tmp" builder) builder)
      | SUnop(op, (((t, _), _) as e)) ->
        let e' = expr builder e in
          (match op with
            A.Neg when t = A.Double -> L.build_fneg e' "tmp" builder
          | A.Neg                   -> L.build_neg e' "tmp" builder
          | A.Not                   -> L.build_not e' "tmp" builder
          | A.Abs when t = A.Double -> L.build_call fabs_func [| e' |] "fabs" builder
          | A.Abs -> L.build_call abs_func [| e' |] "abs" builder
          | A.Transpose when t = A.Matrix -> let original = L.build_load e' "tmp" builder
in
                                let m = L.array_length (L.type_of original) and n =
L.array_length (L.type_of (L.build_load (L.build_gep e' [|L.const_int i32_t 0; L.const_int
i32_t 0|] "tmp" builder) "indexing"  builder)) in
                                let row_idxs = range 0 (n - 1) in
                            let col_idxs = range 0 (m-1) in
                             let matrix = L.build_malloc (array_t n m) "res" builder in
                            List.iter (fun row_idx -> List.iter (fun idx ->
ignore(L.build_store (extract_element e' idx row_idx builder) (L.build_gep matrix
[|L.const_int i32_t 0; L.const_int i32_t row_idx; L.const_int i32_t idx|]  "tmp" builder)
builder)) col_idxs; ) row_idxs; matrix
          | _ -> raise (Failure "internal error: semant should have rejected."))
      | SCall ("print", [e]) -> let expr_value = expr builder e in (match e with (((typ, _),
_) : sexpr) ->
           (match typ with
         | A.Matrix -> let matrix = L.build_load expr_value "tmp" builder in
                                let m = L.array_length (L.type_of matrix) and n =
L.array_length (L.type_of (L.build_load (L.build_gep expr_value [|L.const_int i32_t 0;
L.const_int i32_t 0|] "tmp" builder) "indexing"  builder)) in
                                let row_idxs = range 0 (m - 1) in
                            let col_idxs = range 0 (n-1) in
                            List.iter (fun row_idx -> List.iter (fun idx ->
ignore(L.build_call printf_func [| matrix_format_str ; (L.build_load (L.build_gep expr_value
[|L.const_int i32_t 0; L.const_int i32_t row_idx; L.const_int i32_t idx|] "tmp" builder)
"indexing"  builder) |] "printf" builder)) col_idxs; ignore(L.build_call printf_func [|
return_format_str |] "printf" builder)) row_idxs; L.const_int i32_t 0
         | A.Int | A.Bool -> L.build_call printf_func [| int_format_str ; (expr_value) |]
```

```
"printf" builder
         | A.Double -> L.build_call printf_func [| double_format_str ; (expr_value) |]
"printf" builder
         | A.String -> L.build_call printf_func [| string_format_str; (expr builder e) |]
"printf" builder
         | A.Void -> raise (Failure ("Void type cannot be printed"))))
      | SCall ("sqrt", [e]) -> L.build_call sqrt_func [| (expr builder e) |] "sqrt" builder
      | SCall ("log", [e]) -> L.build_call log_func [| (expr builder e) |] "log" builder
      | SCall ("fill", [(_, row); (_, col); value]) -> (match (row, col) with
        (SIntLit m, SIntLit n) -> let v = expr builder value in
                        let matrix = L.build_malloc (array_t m n) "res" builder in
                           let row_idxs = range 0 (m - 1) in
                          let col_idxs = range 0 (n-1) in
                          List.iter (fun row_idx -> List.iter (fun idx ->
ignore(L.build_store v (L.build_gep matrix [|L.const_int i32_t 0; L.const_int i32_t row_idx;
L.const_int i32_t idx|]  "tmp" builder)  builder)) col_idxs; ) row_idxs; matrix
        | _ -> raise (Failure ("The first two arguments of function fill must be integer")))
      | SCall ("det", [e]) -> let e' = expr builder e in
          let m = (get_matrix_row e' builder) and n = (get_matrix_col e' builder) in
          let matrix_pointer = L.pointer_type (array_t m n) in
          let determinant_t : L.lltype = L.function_type double_t [| matrix_pointer; i32_t;
i32_t |] in
          let determinant_func : L.llvalue = L.declare_function "determinant" determinant_t
the_module in
          if m <> n then raise (Failure ("Cannot compute determinant for given matrix. Row
size and col size must be same."))
          else L.build_call determinant_func [| e'; (L.const_int i32_t m); (L.const_int
i32_t n)|] "determinant" builder
      | SCall ("max_eigvalue", [e]) -> let e' = expr builder e in
          let m = (get_matrix_row e' builder) and n = (get_matrix_col e' builder) in
          let matrix_pointer = L.pointer_type (array_t m n) in
          let max_eigvalue_t : L.lltype = L.function_type double_t [| matrix_pointer; i32_t;
i32_t |] in
          let max_eigvalue_func : L.llvalue = L.declare_function "max_eigvalue"
max_eigvalue_t the_module in
          if m <> n then raise (Failure ("Codegen: Cannot compute the max eigvalue for given
matrix."))
          else L.build_call max_eigvalue_func [| e'; (L.const_int i32_t m); (L.const_int
i32_t n)|] "max_eigvalue" builder
      | SCall ("norm1", [e]) -> let e' = expr builder e in
          let m = (get_matrix_row e' builder) and n = (get_matrix_col e' builder) in
          let matrix_pointer = L.pointer_type (array_t m n) in
          let norm1_t : L.lltype = L.function_type double_t [| matrix_pointer; i32_t; i32_t
|] in
          let norm1_func : L.llvalue = L.declare_function "norm1" norm1_t the_module in
          L.build_call norm1_func [| e'; (L.const_int i32_t m); (L.const_int i32_t n)|]
"norm1" builder
      | SCall ("norm2", [e]) -> let e' = expr builder e in
          let m = (get_matrix_row e' builder) and n = (get_matrix_col e' builder) in
          let matrix_pointer = L.pointer_type (array_t m n) in
```

```
          let norm2_t : L.lltype = L.function_type double_t [| matrix_pointer; i32_t; i32_t
|] in
          let norm2_func : L.llvalue = L.declare_function "norm2" norm2_t the_module in
          L.build_call norm2_func [| e'; (L.const_int i32_t m); (L.const_int i32_t n)|]
"norm2" builder
      | SCall ("tr", [e1]) -> let e1' = expr builder e1  in
          let m = (get_matrix_row e1' builder) and n = (get_matrix_col e1' builder) in
          let result = (L.build_alloca (array_t 1 1) "res" builder) in
            if m <> n then raise (Failure ("Cannot compute trace for given matrix. Row size
and col size must be same."))
           else (
            ignore(L.build_store (L.const_float double_t 0.0) (get_element_address result 0
0 builder) builder);
            for i = 0 to m - 1 do
              ignore(L.build_store (L.build_fadd (extract_element e1' i i builder)
(extract_element result 0 0 builder) "tmp" builder) (get_element_address result 0 0 builder)
builder);
            done;
            (extract_element result 0 0 builder))
      | SCall ("sum_row", [e1]) -> let e1' = expr builder e1  in
          let m = (get_matrix_row e1' builder) and n = (get_matrix_col e1' builder) in
          let result = (L.build_malloc (array_t m 1) "res" builder) in
            for i = 0 to m - 1 do
                ignore(L.build_store (L.const_float double_t 0.0) (get_element_address
result i 0 builder) builder);
                for j = 0 to n - 1 do
                  ignore(L.build_store (L.build_fadd (extract_element e1' i j builder)
(extract_element result i 0 builder) "tmp" builder) (get_element_address result i 0 builder)
builder);
                done
            done;
            result
      | SCall ("sum_col", [e1]) -> let e1' = expr builder e1  in
          let m = (get_matrix_row e1' builder) and n = (get_matrix_col e1' builder) in
          let result = (L.build_malloc (array_t 1 n) "res" builder) in
          for j = 0 to n - 1 do
              ignore(L.build_store (L.const_float double_t 0.0) (get_element_address result
0 j builder) builder);
              for i = 0 to m - 1 do
                ignore(L.build_store (L.build_fadd (extract_element e1' i j builder)
(extract_element result 0 j builder) "tmp" builder) (get_element_address result 0 j builder)
builder);
              done
          done;
          result
      | SCall ("sizeof_row", [e1]) -> let e1' = expr builder e1  in
          let m = (get_matrix_row e1' builder) in
          let result = (L.build_alloca (L.array_type (L.array_type i32_t 1) 1) "res"
builder) in
          ignore(L.build_store (L.const_int i32_t m) (get_element_address result 0 0
```

```
builder) builder);
          (extract_element result 0 0 builder)
      | SCall ("sizeof_col", [e1]) -> let e1' = expr builder e1  in
          let m = (get_matrix_col e1' builder) in
          let result = (L.build_alloca (L.array_type (L.array_type i32_t 1) 1) "res"
builder) in
          ignore(L.build_store (L.const_int i32_t m) (get_element_address result 0 0
builder) builder);
          (extract_element result 0 0 builder)
      | SCall ("mean_row", [e1]) -> let e1' = expr builder e1  in
          let m = (get_matrix_row e1' builder) and n = (get_matrix_col e1' builder) in
          let result = (L.build_malloc (array_t m 1) "res" builder)  and len =
(L.build_alloca double_t "s" builder) in
            for i = 0 to m - 1 do
                ignore(L.build_store (L.const_float double_t 0.0) (get_element_address
result i 0 builder) builder);
                ignore(L.build_store (L.const_float double_t 0.0) len builder);
                for j = 0 to n - 1 do
                  ignore(L.build_store (L.build_fadd (extract_element e1' i j builder)
(extract_element result i 0 builder) "tmp" builder) (get_element_address result i 0 builder)
builder);
                  ignore(L.build_store (L.build_fadd (L.build_load len "len" builder)
(L.const_float double_t 1.0) "add" builder) len builder);
                done;
                ignore(L.build_store (L.build_fdiv (extract_element result i 0 builder)
(L.build_load len "1" builder) "div" builder) (get_element_address result i 0 builder)
builder);
            done;
            result
      | SCall ("mean_col", [e1]) -> let e1' = expr builder e1  in
          let m = (get_matrix_row e1' builder) and n = (get_matrix_col e1' builder) in
          let result = (L.build_malloc (array_t 1 n) "res" builder) and len =
(L.build_alloca double_t "s" builder) in
            for j = 0 to n - 1 do
                ignore(L.build_store (L.const_float double_t 0.0) (get_element_address
result 0 j builder) builder);
                ignore(L.build_store (L.const_float double_t 0.0) len builder);
                for i = 0 to m - 1 do
                  ignore(L.build_store (L.build_fadd (extract_element e1' i j builder)
(extract_element result 0 j builder) "tmp" builder) (get_element_address result 0 j builder)
builder);
                  ignore(L.build_store (L.build_fadd (L.build_load len "len" builder)
(L.const_float double_t 1.0) "add" builder) len builder);
                done;
                ignore(L.build_store (L.build_fdiv (extract_element result 0 j builder)
(L.build_load len "1" builder) "tmp" builder) (get_element_address result 0 j builder)
builder);
            done;
           result
      | SCall ("inv", [e]) -> let e' = expr builder e in
```

```
          let m = (get_matrix_row e' builder) and n = (get_matrix_col e' builder) in
          if m <> n then raise (Failure ("Codegen: Cannot compute inverse for given matrix.
Row size and col size must be same."))
          else (
          let matrix_pointer = L.pointer_type (array_t m n) in
          let inverse_t : L.lltype = L.function_type matrix_pointer [| matrix_pointer;
i32_t|] in
          let inverse_func : L.llvalue = L.declare_function "inverse" inverse_t the_module
in
          L.build_call inverse_func [| e'; (L.const_int i32_t m)|] "inverse" builder)
      | SCall (f, args) ->
        let (fdef, fdecl) = StringMap.find f function_decls in
           let cast value ((t, _), _) =
              (match t with
              A.Matrix -> L.build_bitcast value (ltype_of_typ t) "tmp" builder
            |_ -> value) in
           let llargs = List.rev (List.map (expr builder) (List.rev args)) in
           let save_to_map e (t, n, _, _) = (match t with
           A.Matrix -> (let e' = expr builder e in
              let row = get_matrix_row e' builder and col = get_matrix_col e' builder in
              Hashtbl.add matrix_size (String.concat "" [fdecl.sfname; n]) (row, col))
          |_ -> ()) in
           let llargs_cast = ignore(List.iter2 save_to_map args fdecl.sargs); List.map2 cast
llargs args in
           let result = (match fdecl.styp with
              A.Void -> ""
            | _ -> f ^ "_result") in
        let result = L.build_call fdef (Array.of_list llargs_cast) result builder in
        (match fdecl.styp with
        A.Matrix -> (L.build_bitcast result (L.pointer_type (array_t g_row g_col)) "tmp"
builder)
        | _ -> result)
      | _ -> raise (Failure ("Codegen: expr not supported"))
    in

(* Create a map of global variables after creating each *)
  let global_var (t, n, (row_size, col_size), ((_, e) as e')) =
    let init = (match e with
    SNoexpr -> (match t with
        A.String -> L.const_string context "0"
      | A.Double -> L.const_float (ltype_of_typ t) 0.0
      | A.Matrix -> L.const_pointer_null (L.pointer_type (array_t row_size col_size))
      | _ -> L.const_int (ltype_of_typ t) 0
    )
    |_ -> expr builder e')
     in Hashtbl.add global_vars n (L.define_global n init the_module) in
  List.iter global_var globals;

let add_formal (t, n, (_, _), _) p =
      let ptr = (match t with
```

```
          A.Matrix ->  let (r, c) = Hashtbl.find matrix_size (String.concat ""
[fdecl.sfname; n]) in
          (L.build_bitcast p (L.pointer_type (array_t r c)) "tmp" builder)
          | _ -> p) in
        L.set_value_name n ptr;
      let local = (match t with
       A.Matrix -> let (r, c) = Hashtbl.find matrix_size (String.concat "" [fdecl.sfname;
n]) in
       L.build_alloca (L.pointer_type (array_t r c)) n builder
       | _ -> L.build_alloca (ltype_of_typ t) n builder) in
        ignore (L.build_store ptr local builder); Hashtbl.add local_vars n local in
        List.iter2 add_formal fdecl.sargs (Array.to_list (L.params the_function));

      (* Allocate space for any locally declared variables and add the
       * resulting registers to our map *)
  let add_local (t, n, (row_size, col_size), ((_, e) as e')) =
  let local_var = (match t with
       A.Matrix -> L.build_alloca (L.pointer_type (array_t row_size col_size)) n builder
       | _ -> L.build_alloca (ltype_of_typ t) n builder)
  in ignore (if e <> SNoexpr then let el = expr builder e' in
       (match t with A.String -> let m = get_matrix_row el builder in
                            let dest = L.build_bitcast local_var (L.pointer_type
(L.pointer_type (L.array_type i8_t m))) "tmp" builder in
                             ignore(L.build_store el dest builder)
                          | _ -> ignore(L.build_store el local_var builder))); Hashtbl.add
local_vars n local_var
      in List.iter add_local fdecl.slocals;

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

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

    let rec stmt builder = function
        SBlock sl -> List.fold_left stmt builder sl
      | SExpr e -> ignore(expr builder e); builder
      | SReturn e -> ignore(match fdecl.styp with
          (* Special "return nothing" instr *)
            A.Void -> L.build_ret_void builder
          (* Build return statement *)
          | A.Matrix -> let e' = expr builder e in
              L.build_ret (L.build_bitcast e' (ltype_of_typ A.Matrix) "tmp" builder) builder
```

```
        | _ -> L.build_ret (expr builder e) builder );
      builder
    | SIf (predicate, then_stmt, else_stmt) ->
       let bool_val = expr builder predicate in
   let merge_bb = L.append_block context "merge" the_function in
        let build_br_merge = L.build_br merge_bb in (* partial function *)

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

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

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

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

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

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

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

      (* Implement for loops as while loops *)
      | SFor (e1, e2, e3, body) -> stmt builder
      ( SBlock [SExpr e1 ; SWhile (e2, SBlock [body ; SExpr e3]) ] )
      | _ -> raise (Failure "Codegen: Other stmts not supported yet.")
    in

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

  (* Add a return if the last block falls off the end *)
  add_terminal builder (match (fdecl.styp, fdecl.ssize) with
      (A.Void, _) -> L.build_ret_void
    | (A.Matrix, (row, col)) -> L.build_ret (L.const_pointer_null (array_t row col))
    | (t, _) -> L.build_ret (L.const_int (ltype_of_typ t) 0))
in

List.iter build_function_body functions;
```

```
  the_module
```

8.7 matrices.c Author: Boya Song and Yuli Han

```c
//Reference: https://www.sanfoundry.com/c-program-find-inverse-matrix/
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

double determinant(double*, int);
double* cofactor(double*, int);
double* transpose(double*, double*, int);
double* inverse(double*, int);

/*For calculating Determinant of the Matrix */
double determinant(double* a, int k)
{
  double s = 1, det = 0;
  double *b;
  int i, j, m, n, c;
  b = malloc(k * k * sizeof(double));
  if (k == 1)
    {
     return (*a);
    }
  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++)
              {
                *(b + i*k + j) = 0;
                if (i != 0 && j != c)
                 {
                   *(b + m*k + n) = *(a + i*k + j);
                   if (n < (k - 2))
                    n++;
                   else
                    {
                     n = 0;
                     m++;
                     }
                  }
              }
```

```
            }
          det = det + s * ((*(a + 0*k + c)) * determinant(b, k - 1));
          s = -1 * s;
          }
    }
    free(b);
    return (det);
}

double norm1(double* numbers, int rows, int columns) {
  double res=0;
  for (int j = 0; j < columns; j ++) {
    double sum = 0;
    for (int i = 0; i < rows; i ++) {
      sum += fabs(numbers[i * columns + j]);
    }
    if (sum > res) res = sum;
  }
  return res;
}

double* matMul(double* a, int rows, int columns, double* b) {
  double* res = malloc(rows*sizeof(double));
  for (int i = 0; i < rows; i ++) {
    res[i] = 0;
    for (int k = 0; k < rows; k ++) {
      res[i] += a[i * columns + k] * b[k];
    }
  }
  return res;
}

double* transpose2(double* numbers, int rows, int columns) {
  double* trans = malloc(rows * columns * sizeof(double));
  for (int i = 0; i < rows; i ++) {
    for (int j = 0; j < columns; j ++) {
      trans[j * rows + i] = numbers[i * columns + j];
    }
  }
  return trans;
}

double max_eigvalue(double* numbers, int rows, int columns) {
  double* initVec = malloc(rows*sizeof(double));
  double max = 0;
  initVec[0] = 1;
  for (int i = 1; i < rows; i ++) initVec[i] = 0;
  for (int j = 0; j < 10000; j ++) {
    max = fabs(initVec[0]);
    for (int i = 1; i < rows; i ++) {
```

```
      if (max < fabs(initVec[i])) max = fabs(initVec[i]);
    }
    if (max != 0) {
      for (int i = 0; i < rows; i ++) {
        initVec[i] = initVec[i] / max;
      }
    }
    initVec = matMul(numbers, rows, columns, initVec);
  }
  return max;
}

double norm2(double* numbers, int rows, int columns) {
  if (rows == 1) {
    double res = 0;
    for (int i = 0; i < columns; i ++) {
      res += numbers[i] * numbers[i];
    }
    return sqrt(res);
  }
  double* trans = transpose2(numbers, rows, columns);
  double* m = malloc(rows * rows * sizeof(double));
  for (int i = 0; i < rows; i ++) {
    for (int j = 0; j < rows; j ++) {
      m[i * rows + j] = 0;
      for (int k = 0; k < columns; k ++) {
        m[i * rows + j] += numbers[i * columns + k] * numbers[k * rows + j];
      }
    }
  }
  return sqrt(max_eigvalue(m, rows, rows));
}

double* cofactor(double *num, int f)
{
 double *b, *fac;
 int p, q, m, n, i, j;
 b = malloc(f * f * sizeof(double));
 fac = malloc(f * f * sizeof(double));
 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 + m*f + n) = *(num + i*f + j);
            if (n < (f - 2))
             n++;
            else
             {
               n = 0;
               m++;
               }
            }
        }
      }
      *(fac + q*f + p) = pow(-1, q + p) * determinant(b, f - 1);
    }
  }
  free(b);
  return transpose(num, fac, f);
}
/*Finding transpose of matrix*/
double* transpose(double *num, double* fac, int r)
{
  int i, j;
  double *b, *inverse;
  double d;

  b = malloc(r * r * sizeof(double));
  inverse = malloc(r * r * sizeof(double));

  for (i = 0;i < r; i++)
    {
     for (j = 0;j < r; j++)
       {
         *(b + i*r + j) = *(fac + j*r + i);
        }
    }
  d = determinant(num, r);
  for (i = 0;i < r; i++)
    {
     for (j = 0;j < r; j++)
       {
        *(inverse + i*r + j) = (*(b + i*r + j)) / d;
        }
    }
    free(b);
    free(fac);
    return inverse;
}

double* inverse(double *m, int k) {
  double* invert;
```

```
  double d = determinant(m, k);
  if (d == 0) {
    printf("\nInverse of Entered Matrix is not possible\n");
    invert = NULL;
  }  else
    invert = cofactor(m, k);
  return invert;
}

#ifdef BUILD_TEST
int main()
{
  double* arr;
  double* invert;
  int i, j, r;
  r = 2;
  arr = malloc(2 * 2 * sizeof(double));
  *(arr + 0) = 1;
  *(arr + 1) = 2;
  *(arr + 2) = 3;
  *(arr + 3) = 4;
  invert = inverse(arr, 2);
   printf("The inverse of matrix is : \n");

   for (i = 0;i < r; i++)
    {
     for (j = 0;j < r; j++)
       {
         printf("\t%lf", *(invert + i*r + j));
        }
    printf("\n");
     }

}
#endif
```

8.7 Makefile Author: Boya Song

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

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

# "make all" builds the executable

.PHONY : all
all : mathlight.native matrices.o
```

```
# Testing the "printbig" example

matrices : matrices.c
	cc -o matrices -DBUILD_TEST matrices.c


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

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

# "make clean" removes all generated files

.PHONY : clean
clean :
	ocamlbuild -clean
	rm -rf testall.log ocamlllvm *.diff
```

8.8 testall.sh Author: Boya Song

```
#!/bin/sh

# Regression testing script for MathLight
# 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 mathlight compiler.  Usually "./mathlight.native"
MATHLIGHT="./mathlight.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] [.txt 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/.txt//'`
    reffile=`echo $1 | sed 's/.txt$//'`
    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
${basename}.out" &&
    Run "$MATHLIGHT" "$1" ">" "${basename}.ll" &&
    Run "$LLC" "-relocation-model=pic" "${basename}.ll" ">" "${basename}.s" &&
    Run "$CC" "-o" "${basename}.exe" "${basename}.s" "matrices.o"&&
    Run "./${basename}.exe" ">" "${basename}.out" &&
    Compare ${basename}.out ${reffile}.out ${basename}.diff

    # Report the status and clean up the generated files

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

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

    echo -n "$basename..."

    echo 1>&2
```

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

    generatedfiles=""

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

    # Report the status and clean up the generated files

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

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

shift `expr $OPTIND - 1`

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

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

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

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

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

exit $globalerror
```

8.9 Failed test cases

8.9.1 fail-assignerr.txt

```
func int main() {
  int a;
  string b;
  b = "123";
  a = b;
  return 0;
}
```

Output:
Fatal error: exception Failure("illegal assignment int(-1, -1) = string(-1, -1) in a = b")

8.9.2 fail-duplicatefuncs.txt

```
func int a(){
    return 1;
}
func int a(){
```

```
    return 2;
}
func int main() {
  return 0;
}
```

Output:
Fatal error: exception Failure("duplicate function a")

8.9.3 fail-duplicatevars.txt

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

Output:
Fatal error: exception Failure("duplicate local_vars a")

8.9.4 fail-forloop.txt

```
func int main() {
  string a = "123";
  int i = 1;
  for(i = 0; a; i = i+1){
      print(i);
  }
  return 0;
}
```

Output:
Fatal error: exception Failure("expected Boolean expression in a")

8.9.5 fail-funcnotexist.txt

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

Output:
Fatal error: exception Failure("unrecognized function myfunc")

8.9.6 fail-funcwrongargslen.txt

```
func int add(int a, int b) {
    return a+b;
}
func int main () {
    return add(1, 2, 3);
}
```

Output:
Fatal error: exception Failure("expecting 2 arguments in add(1, 2, 3)")

8.9.7 fail-funcwrongargstype.err

```
func int add(int a, int b) {
    return a+b;
}
func int main () {
    return add(1, "123");
}
```

Output:
Fatal error: exception Failure("illegal argument found string 123 expected int in add(1, 123)")

8.9.8 fail-funcwrongreturntype.txt

```
func double add(int a, int b) {
    return a+b;
}
func int main () {
    return add(1,2);
}
```

Output:
Fatal error: exception Failure("return gives double expected int in add(1, 2)")

8.9.9 fail-helloworld.txt

```
print("Hello World!");
```

Output:
Fatal error: exception Stdlib.Parsing.Parse_error

8.9.10 fail-ifwrongcond.txt

```
func int main() {
  string a = "123";
  if(a){
      return 1;
  }
  return 0;
}
```

Output:
Fatal error: exception Failure("expected Boolean expression in a")

8.9.11 fail-illegalbinop.txt

```
func int main() {
  int a;
  string b;
  return a + b;
}
```

Output:
Fatal error: exception Failure("illegal binary operator: int of size (-1,-1) + string of size (-1,-1) in a + b")

8.9.12 fail-illegalcharacter.txt

```
func int main() {
  int a;
  string b;$
}
```

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

8.9.13 fail-illegalmatrixbinop.txt

```
func int main() {
  matrix a <2,2>;
  matrix b <3,3>;
  print(a+b);
  return 0;
}
```

Output:

Fatal error: exception Failure("illegal binary operator: matrix of size (2,2) + matrix of size (3,3) in a + b")

8.9.14 fail-matrixaccess.txt

```
func int main() {
  matrix a <3, 3>;
  a = [1.1, 2.1, 3.1; 1.0, 2.0, 3.0; 4.1, 4.2, 4.3];
  print(a[3,3]);
  return 0;
}
```

Output:
Fatal error: exception Failure("expression SMatrix2DElement a[3, 3] out of boundary, matrix size: (3, 3")

8.9.15 fail-matrixaccesstype.txt

```
func int main() {
  matrix a <3, 3>;
  a = [1.1, 2.1, 3.1; 1.0, 2.0, 3.0; 4.1, 4.2, 4.3];
  print(a[1.1,2.2]);
  return 0;
}
```

Output:
Fatal error: exception Failure("illegal 2D matrix index type [double, double] for matrix a")

8.9.16 fail-matrixconcat.txt

```
func int main () {
    matrix i <3, 3>;
    matrix j <2, 2>;
    i = [1.0,2.0,3.0;1.0,2.0,3.0];
    j = [4.0,5.0;4.0,5.0];
    print(i);
    print([i; j]);
    return 0;
}
```

Output:
Fatal error: exception Failure("illegal Matrix Concat operator: matrix of size (3,3) : matrix of size (2,2) in MatrixOp i:j")

8.9.17 fail-matrixmodifytype.txt

```
func int main() {
  matrix a <3, 3>;
  a = [1.1, 2.1, 3.1; 1.0, 2.0, 3.0; 4.1, 4.2, 4.3];
  a[1,1] = "123";
  return 0;
}
```

Output:
Fatal error: exception Failure("illegal 2D matrix assignment, number expected, get string")

8.9.18 fail-nomain.txt

```
func int a() {
    return 0;
}
```

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

8.9.19 fail-noreturn.txt

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

Output:
Fatal error: exception Failure("function a has no return")

8.9.20 fail-varnoexist.txt

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

Output:
Fatal error: exception Failure("undeclared identifier a")

8.9.21 fail-wrongrange.txt

```
func matrix main () {
    matrix i <3, 3>;
    i = [1.0,2.0,3.0;1.0,2.0,3.0];
    return i[2:1,1];
}
```

Output:
Fatal error: exception Failure("Invalid argument in Range: 2: 1")

8.9.21 fail-wrongrangetype.txt

```
func matrix main () {
    matrix i <3, 3>;
    i = [1.0,2.0,3.0;1.0,2.0,3.0];
    return i[1.0:2,1];
}
```

Output:
Fatal error: exception Failure("illegal Range type, int : int expected, get double : int in 1. : 2")

8.10 Succeeded test cases

8.10.1 test-helloworld.txt

```
func int main() {
  string s = "Hello World!";
  print(s);
  return 0;
}
```

Output:
Hello World!

8.10.2 test-printbool.txt

```
func int main() {
  boolean a;
  a = True;
  print(a);
  return 0;
}
```

Output: 1

8.10.3 test-printdouble.txt

```
func int main() {
  print(0.35);
  return 0;
}
```

Output: 0.35

8.10.4 test-printint.txt

```
func int main() {
  int a;
  a = -12;
  print(a);
  return 0;
}
```

Output: -12

8.10.5 test-printmatrix.txt

```
func int main() {
  matrix a <3, 3>;
  a = [1.1, 2.1, 3.1; 1.0, 2.0, 3.0; 4.1, 4.2, 4.3];
  print(a);
  return 0;
}
```

Output:
1.1 2.1 3.1
1 2 3
4.1 4.2 4.3

8.10.6 test-variable.txt

```
int i;
double j;

func int main() {
```

```
  string s;
  print("Hello World!");
  return 0;
}
```

Output:
Hello World!

8.10.7 test-integer_operator.txt

```
func int main() {
    int c;
    boolean a;
    c = 1 + 2;
    print(c);
    c = 1 - 2;
    print(c);
    c = 5 * 4;
    print(c);
    c = 6 / 5;
    print(c);
    a = c ==2;
    print(a);
    return 0;
}
```

Output:
3
-1
20
1
0

8.10.8 test-printmatrix_lit.txt

```
func int main() {
  print([1.1, 2.1, 3.1; 1.0, 2.0, 3.0; 4.1, 4.2, 4.3]);
  return 0;
}
```

Output:
1.1 2.1 3.1
1 2 3

4.1 4.2 4.3

8.10.9 test-double_operator.txt

```
func int main() {
    double c;
    c = 1.3 + 2.4;
    print(c);
    return 0;
}
```

Output:
3.7

8.10.10 test-double_mul_matrix.txt

```
func int main() {
    double c;
    matrix m<2,2>;
    matrix res<2,2>;
    c = 3.0;
    m = [1.0,1.0;1.0,1.0];
    res = c * m;
    print(res);
    res = m * c;
    print(res);
    return 0;
}
```

Output:
3 3
3 3
3 3
3 3

8.10.11 test-fill.txt

```
func int main() {
  matrix a <2, 3>;
  a = fill(2, 3, 1.1);
  print(a);
  return 0;
```

```
}
```

Output:
1.1 1.1 1.1
1.1 1.1 1.1

8.10.12 test-sqrt.txt

```
func int main() {
  double a;
  a = 16.0;
  print(sqrt(a));
  return 0;
}
```

Output:
4

8.10.13 test-func.txt

```
func double add(double a, double b) {
  return a + b;
}
func int main() {
  double a;
  a = add(1.0, 2.0);
  print(a);
  return 0;
}
```

Output:
3

8.10.14 test-matrixaccess.txt

```
func int main() {
  matrix a <3, 3>;
  double b;
  a = [1.1, 2.1, 3.1; 1.0, 2.0, 3.0; 4.1, 4.2, 4.3];
  b = a[0,0] + a[1,2];
```

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

Output:
4.1

8.10.15 test-mat_add.txt

```
func int main() {
    matrix a<2,2>;
    matrix b<2,2>;
    matrix res<2,2>;
    a = [1.0,2.0;3.0,4.0];
    b = [4.0,3.0;2.0,1.0];
    res = a + b;
    print(res);
    return 0;
}
```

Output:
5 5
5 5

8.10.16 test-mat_sub.txt

```
func int main() {
    matrix a<2,2>;
    matrix b<2,2>;
    matrix res<2,2>;
    a = [1.0,2.0;3.0,4.0];
    b = [4.0,3.0;2.0,1.0];
    res = a - b;
    print(res);
    return 0;
}
```

Output:
-3 -1
1 3

8.10.17 test-mat_mul.txt

```
func int main() {
    matrix a<2,2>;
    matrix b<2,2>;
    matrix res<2,2>;
    a = [1.0,2.0;3.0,4.0];
    b = [4.0,3.0;2.0,1.0];
    res = a * b;
    print(res);
    return 0;
}
```

Output:
8 5
20 13

8.10.18 test-mat_concat_ver.txt

```
func int main() {
    matrix a<2,2>;
    matrix b<1,2>;
    matrix res<3,2>;
    a = [1.0,2.0;3.0,4.0];
    b = [4.0,3.0];
    res = [a;b];
    print(res);
    return 0;
}
```

Output:
1 2
3 4
4 3

8.10.19 test-mat_concat_hor.txt

```
func int main() {
    matrix a<2,2>;
    matrix b<2,1>;
    matrix res<2,3>;
    a = [1.0,2.0;3.0,4.0];
    b = [4.0;3.0];
```

```
    res = [a,b];
    print(res);
    return 0;
}
```

Output:
1 2 4
3 4 3

8.10.20 test-neg.txt

```
func int main() {
  int a;
  int b;
  a = 1;
  b = -a;
  print(a);
  print(b);
  return 0;
}
```

Output:
1
-1

8.10.21 test-not.txt

```
func int main() {
  boolean a;
  boolean b;
  a = True;
  b = !a;
  print(a);
  print(b);
  return 0;
}
```

Output:
1
0

8.10.22 test-transpose.txt

```
func int main() {
  matrix a <2, 3>;
  matrix b <3, 2>;
  a = [1.1, 2.1, 3.1; 1.0, 2.0, 3.0];
  b = a';
  print(b);
  return 0;
}
```

Output:
1.1 1
2.1 2
3.1 3

8.10.23 test-if.txt

```
func int main() {
  if (1>2) print("first");
  else if (3>4) print("second");
  else print("final");
  return 0;
}
```

Output:
Final

8.10.24 test-for.txt

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

Output:
0
1
2
3

4

8.10.25 test-while.txt

```
func int main() {
  int i;
  i = 0;
  while (i < 5) {
     print(i);
     i = i + 1;
  }
  return 0;
}
```

Output:
0
1
2
3
4

8.10.26 test-matrix_dot_mul.txt

```
func int main() {
    matrix a<3,2>;
    matrix b<3,2>;
    matrix res<3,2>;
    a = [1.0,1.0;1.0,1.0;1.0,1.0];
    b = [1.5,2.0;2.0,1.5;4.0,3.0];
    res = a .* b;
    print(res);
    res = b .* a;
    print(res);
    return 0;
}
```

Output:
1.5 2
2 1.5
4 3
1.5 2
2 1.5

4 3

8.10.27 test-matrix_dot_div.txt

```
func int main() {
    matrix a<3,2>;
    matrix b<3,2>;
    matrix res<3,2>;
    a = [1.0,1.0;1.0,1.0;1.0,1.0];
    b = [2.0,2.0;2.0,4.0;4.0,2.0];
    res = a ./ b;
    print(res);
    res = b ./ a;
    print(res);
    return 0;
}
```

Output:
0.5 0.5
0.5 0.25
0.25 0.5
2 2
2 4
4 2

8.10.28 test-determinant.txt

```
func int main() {
  matrix a <2, 2>;
  double d;
  a = [3.0, 8.0; 4.0, 6.0];
  d = det(a);
  print(d);
  return 0;
}
```

Output:
-14

8.10.29 test-func-inv.txt

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

```
  matrix a <2, 2>;
  matrix d <2, 2>;
  a = [1.0, 2.0; 3.0, 4.0];
  d = inv(a);
  print(d);
  return 0;
}
```

Output:
-2 1
1.5 -0.5

8.10.30 test-sum.txt

```
func int main() {
  matrix a <3, 3>;
  matrix sum_r<3,1>;
  matrix sum_c<1,3>;
  a = [1.1, 1.2, 1.3; 1.0, 1.4, 1.0; 1.0, 1.0, -1.0];
  sum_r = sum_row(a);
  print(sum_r);
  sum_c = sum_col(a);
  print(sum_c);
  return 0;
}
```

Output:
3.6
3.4
1
3.1 3.6 1.3

8.10.31 test-mean.txt

```
func int main() {
  matrix a <3, 3>;
  matrix mean_r<3,1>;
  matrix mean_c<1,3>;
  a = [1.1, 1.2, 1.3; 1.0, 1.3, 1.0; 1.0, 1.0, 1.0];
  mean_r = mean_row(a);
  print(mean_r);
  a = [1.1, 1.0, 1.0; 1.2, 1.3, 1.0; 1.3, 1.0, 1.0];
  mean_c = mean_col(a);
```

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

Output:
1.2
1.1
1
1.2 1.1 1

8.10.32 test-func-matrix-return.txt

```
func matrix add (matrix a, matrix b) {
  matrix c <2,2> = a + b;
  print(c);
  return c;
}

func int main(){
  matrix a <2, 2>;
  matrix b <2, 2>;
  a = [1, 2; 3, 4];
  b = [0, 1; 1, 0];
  print(a);
  print(b);
  print(add(a, b));
  return 0;
}
```

Output:
1 2
3 4
0 1
1 0
1 3
4 4
1 3
4 4

8.10.33 test-lr-calc.txt

```
func matrix calculate_weight (matrix x, matrix y) {
  matrix result <2, 1>;
  result = inv(x' * x) * x' * y;
  return result;
}

func void gradient_descent (matrix x, matrix y) {
  int n = 3;
  matrix y_current <3, 1>;
  matrix sum_c <1,1>;

  double learning_rate = 0.0001;
  double w_current = 0.0;
  double b_current = 0.0;
  double w_gradient = 0.0;
  double b_gradient = 0.0;

  int i;
  int j;
  double cost;

  for (i = 0; i < 250000; i = i + 1) {
    y_current = (w_current * x) + b_current;
    cost = 0.0;
    for(j = 0; j < n; j = j + 1){
           cost = cost + (y[j,0] - y_current[j,0]) * (y[j,0] - y_current[j,0]);
    }
    cost = cost / n;
    sum_c = sum_col(x .* (y - y_current));
    w_gradient = -(2.0/n) * sum_c[0,0];
    sum_c = sum_col(y - y_current);
    b_gradient = -(2.0/n) * sum_c[0,0];
    w_current = w_current - (learning_rate * w_gradient);
    b_current = b_current - (learning_rate * b_gradient);
  }
  print("Approach 2: Solving equation by gradient descent:");
  print("After 250000 iterations, the final cost is");
  print(cost);
  print("W is");
  print(w_current);
  print(b_current);
}

func int main() {
  matrix x <3, 1> = [1;2;3];
  matrix y <3, 1> = [4;5;6];
  matrix ones <3, 1> = fill(3, 1, 1.0);
  matrix x_new <3, 2>;
  matrix w <2, 1>;
  matrix y_lr <3, 1>;
```

```
  /* analytical solution for linear regression */
  /* x_pre * w_pre + b = y
     [x_pre : ones] * w = y
     x * w = y
     x' * x * w = x' * y
     w = inv(x' * x) * x' * y */
  x_new = [x, ones];
  w = calculate_weight(x_new, y);
  y_lr = x_new * w;

  print("Given X and Y, linear regression is finding a w so that Y = W * X");
  print("Approach 1: Solving equation analytically:");
  print("W is");
  print(w);
  print("The predicted y is:");
  print(y_lr);

  /* gradient descent for linear regreesion */
  gradient_descent(x, y);
  return 0;
}
```

Output:
Given X and Y, linear regression is finding a w so that Y = W * X
Approach 1: Solving equation analytically:
W is
1
3
The predicted y is:
4
5
6
Approach 2: Solving equation by gradient descent:
After 250000 iterations, the final cost is
3.97618e-06
W is
1.00232
2.99474
8.10.34 test-double_add_mat.txt

```
func int main() {
  matrix a <2, 2>;
```

```
  double b;
  a = [1.0, 2.0; 3.0, 4.0];
  b = 2.0;
  print(b * a);
  print(a * b);
  return 0;
}
```

Output:
2 4
6 8
2 4
6 8

8.10.35 test-demo2.txt

```
func int main(){
  matrix col1 <2, 1> = [3.0; 1.0];
  matrix col2 <2, 1> = [2.0; 4.0];
  matrix arr <2,2> = [col1, col2];
  int a = 1;
  double b = -arr[0,0] - arr[1,1];
  double c = arr[0,0] * arr[1,1] - arr[0,1] * arr[1,0];
  double eigv1 = ( -b + sqrt(b*b - 4 * a * c)) / (2 * a);
  double eigv2 = ( -b - sqrt(b*b - 4 * a * c)) / (2 * a);

  print("Calculate eigenvalue:");
  print(eigv1);
  print(eigv2);
  return 0;
}
```

Output:
Calculate eigenvalue:
5
2

8.10.36 test-double_ope_int.txt

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

```
  a = -1.1;
  b = 2;
  c = a + b;
  print(c);
  c = a - b;
  print(c);
  c = a * b;
  print(c);
  c = a / b;
  print(c);
  return 0;
}
```

Output:
0.9
-3.1
-2.2
-0.55

8.10.37 test-mat_ope_int.txt

```
func int main(){
  matrix a<2,2>;
  int b;
  matrix res<2,2>;
  a = [1,2;3,4];
  b = 2;
  res = a + b;
  print(res);
  res = b + a;
  print(res);
  res = a * b;
  print(res);
  res = b * a;
  print(res);
  return 0;
}
```

Output:
3 4
5 6
3 4
5 6
2 4

6 8
2 4
6 8

8.10.38 test-tr.txt

```
func int main(){
  matrix a<2,2>;
  a = [1.0,2.0;-2.0,-2.0];
  print(tr(a));
  return 0;
}
```

Output:
-1

8.10.39 test-element-change.txt

```
func int main() {
  matrix a <3, 3>;
  double b;
  a = [1.1, 2.1, 3.1; 1.0, 2.0, 3.0; 4.1, 4.2, 4.3];
  b = a[0,0] + a[1,2];
  print(b);
  a[0, 0] = 2.3;
  a[1, 2] = 2.1;
  print(a[0, 0] + a[1, 2]);
  return 0;
}
```

Output:
4.1
4.4

8.10.40 test-func-matrix.txt

```
func void test(matrix a, matrix b) {
  matrix c <3, 3> = a + b;
  print(a);
  print(c);
}
```

```
func int main() {
  matrix a <3, 3>;
  matrix b <3, 3>;
  a = [1.1, 2.1, 3.1; 1.0, 2.0, 3.0; 4.1, 4.2, 4.3];
  b = [1.1, 2.1, 3.1; 1.0, 2.0, 3.0; 4.1, 4.2, 4.3];
  test(a, b);
  return 0;
}
```

Output:
1.1 2.1 3.1
1 2 3
4.1 4.2 4.3
2.2 4.2 6.2
2 4 6
8.2 8.4 8.6

8.10.41 test-norm2.txt

```
func int main() {
  matrix a <3, 3> = [1.1, 2.1, 3.1; 1.0, 2.0, 3.0; 4.1, 4.2, 4.3];
  matrix b<2,3> = [1,2,3;4,5,6];
  matrix c<3> = [1,2,3];
  print(norm2(a));
  print(norm2(b));
  print(norm2(c));
  return 0;
}
```

Output:
8.75668
9.25091
3.74166

8.10.42 test-abs.txt

```
func int main(){
  int a = -3;
  double b = 4.2;
  print(|a|);
  print(|b|);
  return 0;
```

```
}
```

Output:
3
4.2

8.10.43 test-comments.txt

```
func int main() {
  // short comment here, long comment below
  /* int i;
  i = 0; */
  print(1);
  return 0;
}
```

Output:
1

8.10.44 test-global-declare-and-assign.txt

```
int a = 1;
int b = 2;
matrix c <2, 2>;
func int main(){
  print(a + b);
  return 0;
}
```

Output:
3

8.10.45 test-log.txt

```
func int main(){
  double b = 2.0;
  print(log(b));
  return 0;
}
```

Output:

0.693147

8.10.46 test-logical-op.txt

```
func int main() {
  int a;
  int b;
  a = 2;
  b = 3;
  if (a > b || a == b) print(a);
  else print(b);
  return 0;
}
```

Output:
3

8.10.47 test-pow.txt

```
func int main(){
  int a = 2;
  double b = 3.0;
  double c = a^b;
  int d = a^a;
  double e = a^b;
  double f = b^a;
  double g = b^b;
  print(d);
  print(e);
  print(f);
  print(g);
  return 0;
}
```

Output:
4
8
9
27
8.10.48 test-primitive-op.txt

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

```
  double b;
  double c;
  double d;
  i = 2 + 3;
  b = 1.5;
  c = 0.5;
  d = b / c;
  print(i);
  print(d);
  return 0;
}
```

Output:

5

3