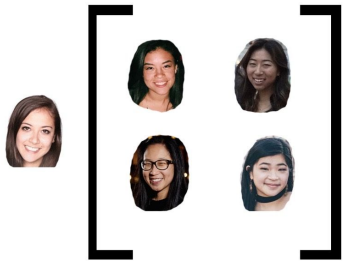
JSTEM: A Matrix Manipulation Language

Final Report

Julia Troxell (jst2152)
Sam Stultz (sks2200)
Tessa Hurr (trh2124)
Emily Song (eks2138)
Michelle Lu (ml3720)



Contents

1. Introduction to the Language

JSTEM is a matrix manipulation language. Matrices are important tools in mathematics with various applications such as graphic rendering and encryption. We have found that matrix manipulation in Java is tedious and difficult. We propose JSTEM, a language that allows for easy and intuitive matrix transformations. JSTEM is compiled into LLVM.

2. Language Tutorial

a. Setup

JSTEM was developed in OCaml. In order to used the compiler, the user should download OCaml. The OCaml Package Manager (OPAM) should be installed to access packages related OCaml. The user should also ensure that the LLVM library that is being installed matches the version of LLVM that is installed on the system of their device.

b. Using the Compiler

In order to use JSTEM, the user should go into the JSTEM directory and run the Makefile by using the command '**`make`**'. This will build the scanner, parser, ast, semant, and codegen files to produce a JSTEM.native file.

To run the full test suite containing the scanner, parser, fail and success tests, use the command "**`make test`**". This will create the JSTEM.native file, test the scanner tests, then the parser tests, and finally the fail and success test files. It will print whether each test fails or succeeds. The fail errors along with the successful commands are printed in the **`testall.log`** file

To test the scanner tests after making JSTEM.native, move to the test directory and run the scanner test script by using the commands:

```
cd tests
./scanner_testing.sh
```

To test the parser tests after making JSTEM.native, move to the test directory if you are not already in it and run the parser test script by using the commands:

```
cd tests
./parser_testing.sh
```

To test either the parser or the scanner if JSTEM.native has not been made, go to the parent directory and run the following commands:

```
make test_parser_scanner
cd tests
make build
./parser_testing.sh
```

To run individual test files, go to the directory where the file exists and run in the JSTEM.native file with the -c flag, filename, and standard library file. The output is written in an output file and then compiled using the lli command. The following is an example.

```
./JSTEM.native -c filename.JSTEM stdlib.JSTEM > output.txt
lli output.txt
```

To test all the fail and success tests, make sure you are in the **`tests`** directory. Run the compiling test script by running the following command. This will also print a fail or success after each test and produce a **`testall.log`** file containing the errors for each failed test.

```
./compiler_testing.sh
```

c. Sample Program

The following is a simple example of a .JSTEM program. All programs contain a main function. The sample.JSTEM file shown below defines a tuple, row, and matrix variable and then initializes each of them. It then prints the first element of each variable.

sample.JSTEM

```
def int main() { /* declaration of a main function */
       int(%3%) tuple_var; /* declaration of a tuple */
       float[4] row_var; /* declaration of a row with size of 4 */
       int[2][2] matrix_var; /* declaration of a 2x2 matrix */

       tuple_var = (% 1, 2, 3 %); /* filling the tuple with int values */
       row_var = [ 4.0, 5.0, 6.0, 7.0 ]; /* filling the row with float values */
       matrix_var = {% 8, 9 | 10, 11%}; /* filling the matrix with int values */

       print(tuple_var(%0%)); /* standard print statement */
       printf(row_var[0]); /* special printf prints out floats */
       print(matrix_var[0][0]);
```

```
        return 0; /* return statement is always needed */
}
```

The output of this program is '**14.0000008**'. The print functions returns the value on the same line without spaces, so the results are printed right next to each other.

3. Language Reference Manual

a. Types

Data Type	Description	Declaration
matrix	A matrix (a list of rows), bars separate rows, can be of type int, float, or tuple	`int[2][2] m;` `m = {% 1, 2 | 3, 4 %};`
row	A list, can be of type int, float, or tuple	`int[3] r;` `r = [1, 2, 3];`
String	String value	`String str = "hello!";`
int	Integer value	`int x;` `x = 5;`
float	Floating point value	`float x;` `x = 5.0;`
File	file	`File file1 =` `open("file.txt","r");`
tuple	Array of length 3, can only be of type int	`int(%3%) t;` `t = (% 1, 2, 3 %);`
bool	value True or False	`bool b = True;`
Row pointer	Point to an element in a row	`int[] x;`
Matrix pointer	Point to an elements in a matrix	`int[[]] x;`

b. Lexical Conventions

b1. Identifiers

An identifier must begin with a lowercase or uppercase character, and can otherwise consist of any combination of these characters along with digits and underscores. Other symbols (i.e. !, @, #, etc.) are prohibited in variable names, and keywords (i.e. for, while, if, length, width, type, etc.) are prohibited as variable names as well. The name _i is also reserved.

b2. Keywords

Keyword	**Description**	**Usage**
for	Iteration until condition is reached	`for(condition) { … }`
while	Loop until condition is reached	`while(condition) { … }`
if	If in an if-else statement	`if { … }`
else	Else in an if-else statement	`else { … }`
return	Returns value from a function	`return m;`
main	Main function	`def void main() { … }`
def	function declaration	`def void x(int y) { … }`

in	Traverse through a sequence in an enhanced matrix for loop	`for(row in matrix){`
		`  ...process row...`
		`}`

void	Return type for a function that does not return anything	`def void no_return() { … }`
length	matrix.length returns the # of rows in the matrix, row.length returns the # of elements in the row	`m = {% 1, 2 | 3, 4 %};` `len = m.length; /* 2 */`
width	matrix.width returns the # of columns in the matrix	`wid = m.width; /* 2 */`
type	(matrix or row).type returns a string	`typ = m.type; /* “int” */`

	representation of the type of each element (either “int”, “float”, or “tuple”)	

b3. Comments

/* … */ Comments start with ‘/*’ and end with ‘*/’.

b4. Operators

Arithmetic Operators	**Description**	**Usage**
,	Separate values in tuple or row	`t = (% 1, 2, 3 %)` `r = [1,2,3]`
:	Specify range of a matrix	`m[2][:]` */* *access row 2 */*
{	Left bound of a function body	`def int main() { ...`
}	Right bound of a function body	`def int main() {` `… }`
{%	Left bound of a matrix	`m = {% 1, 2 | 3, 4 %}`
%}	Right bound of a matrix	`m = {% 1, 2 | 3, 4 %}`
|	Row separator in a matrix	`m = {% 1, 2 | 3, 4 %}`
[	Left bound of a row	`r = [1,2,3]`
]	Right bound of a row	`r = [1,2,3]`
(	Left parand	`print( … )`
)	Right parand	`print( … )`
(%	Left bound of a tuple	`t = (% 1, 2, 3 %)`
%)	Right bound of a tuple	`t = (% 1, 2, 3 %)`
“	Used around strings	`String str = “hello”`
=	Assignment operator	`y = 6, z = 2`
@=	Memory assignment operator	`String line;`

		`line @= new[1000];`
+	Arithmetic operators	`x = y + z` */* x = 8 */*
-	Subtraction operator	`x = y - z` */* x = 4 */*
*	Multiplication operator	`x = y * z` */* x = 12 */*
/	Division operator	`x = y / z` */* x = 3 */*
==	Returns 1 if the values are equal, 0 otherwise	`y == z` */* return 0 */*
!=	Returns 1 if the values are not equal, 0 otherwise	`y != z` */* return 1 */*
>	Greater than operator	`y > z` */* return 1 */*
<	Less than operator	`y < z` */* return 0 */*
>=	Greater than or equal to operator	`y >= z` */* return 1 */*
<=	Less than or equal to operator	`y <= z` */* return 0 */*
&&	Logical AND operator	`0 && 1` */* return 0 */*
||	Logical OR operator	`0 || 1` */* return 1 */*
!	Logical NOT operator	`!(5 == 5)` */* return 0 */*
$	Reference a row ($$ references a matrix)	`int[] r_ptr;` `r_ptr = $r;`
#	Dereference a row or matrix	`int i;` `i = #r_ptr;`
~~	Increment a pointer	`r_ptr = ~~r_ptr;`

Tuple Operator	**Description**	**Usage**
+, -	Tuple addition and subtraction	`t = t1`$_1$ `+ t`$_2$ */*tuple addition */*
+,-,*, /	Tuple scalar operations	`t = 5 * t1`$_1$ */* scalar multiplication */*

Matrix Operator	Description	Usage
+, -	Matrix addition and subtraction	`M = M1 - M2` */* matrix subtraction*/*
+, -, *, /	Matrix scalar operations	`M = M1 / 2` */* scalar division*/*

b5. Precedence

Highest
function and matrix declarations
!
*, /
+, -
<, >, <=, >=
==, !=
&&
||
=
Lowest

c. Built-in Functions

Function	Description	Usage
print(int x)	Prints an int	`print(10);` /* 10 */
printf(float x)	Prints a float	`printf(1.2);` /* 1.2 */
prints(String x)	Prints a string, no newline	`prints("hi");` /* hi */

printfs(String x)	Prints a string with newline	`printfs("hi");`
		`printfs("hello");`
		/* hi
		hello */

printb(Bool x)	Prints a boolean and returns 1 or 0 for True and False	`printb(True);` /* 1 */
open(String x, String y)	Opens the file which is read as a string. Returns a pointer to the file.	`File in_file;` `File out_file;`

	Options for y include: “r”: open the file for reading “r+”: read and write “w”: construct a new file for writing and clear the old file “w+”: construct a new file for reading and writing “a”: append	`in_file = open(“file.txt”,”r”); out_file = open(“file_o.txt”,”w”);`
read(String x, int w, int y, String z)	Reads an entire file Z into X. w is size in bytes of each element to be read and y is number of elements.	`File in_file; String p; in_file = open(“file.txt”,”r”); read(p, 10, 10, in_file);`

write(String x, int y, int w, String z)	Writes to a file z, with contents of string x. y is the size of bytes in each element and w is the number of elements.	`File out_file; String p;`
		
		`p = “hello”; out_file = open(“file_o.txt”,”w”);`
		
		`write(p, 1, len(p), file_out);`

close(File x)	Closes the file	`close(in_file);`
fget(String x, int y, String z)	Reads one line (until new line /n) or y bytes (whichever is shorter) from String Z (File). Line is stored in x. Returns null if nothing is read	`fget(line, 10, file_in);`
atoi(String x)	Converts from an int to a string	`int num; num = atoi(“3”); print(num); /* 3 */`

itos(String x, String y, String z)	Converts from a string to an int. x is the destination, z is the string to convert and y i the format that z should be printed in	`String num; String temp;`
		
		`temp @= new[4]; num = “1234”;`
		
		`itos(temp, “%d”, num); printfs(temp);`

		/* 1234 */

splitstr(String x, String y)	Parses a string x by a delimiter y.	String delim;
		String line;
		String temp;
		File file_in;
		
		line @= new[1000];
		temp @= new[1000];
		delim @= new[1];
		
		delim = ",";
		file_in =
		open("matrix.txt","r");
		fget(line, 10, file_in);
		temp =
		splitstr(line,delim);

find(String x, String y)	Returns a pointer to the first appearance of y in x. Returns null if not found.	String str; str = find(“”,”x”); /* str is null */
strcmp(String x, String y)	Compares strings x and y. Returns an integer. 0 if equal.	String x; String y; x = “hello”; y = “hello”; print(strcmp(x,y)); /* 0 */

NULL(String x)	Returns True if x is null. False otherwise.	String str;
		str = find(“”,”x”);
		if (NULL(str)){
		printfs(“IS NULL”);
		}
		/* IS NULL */

d. Standard Library Functions

Function	**Description**	**Usage**
print_tuple(int(%3%) t)	Prints a tuple of length 3, argument is the tuple being	t1 = (%1, 2, 3%) print_tuple(t1); // (1,2,3)

	printed	
print_rowi(int[] r, int len)	Prints a row of ints, pass in pointer to the int row and row length as arguments	`r = [2,3,5,6];` `print_rowi($r,4); // [2,3,4,6]`
print_rowf(float[] r, int len)	Prints a row of floats, pass in pointer to the float row and row length as arguments	`r = [1.0, 2.1];` `print_rowf($r, 2); /* [1.0, 2.1] */`

print_rowt(int(%3%)[] r, int len)	Prints row of tuples, pass in pointer to the tuple row and row length as arguments	`r = [(%1,1,1%),(%2,2,2%)];`
		`print_rowt($r, 2);`
		`/* [(1,1,1), (2,2,2)] */`

print_matrixi(int[[]] m, int len, int wid)	Prints matrix of ints, pass in pointer to int matrix, length, and width as arguments	`m = {% 1, 2 | 3, 4 %};`
		`print_matrixi($$m,2,2);`
		`/* { 1, 2`
		`3, 4 } */`

print_matrixf(float[[]] m, int len, int wid)	Prints matrix of floats, pass in pointer to float matrix, length, and width as arguments	`m = {% 1.1, 2.2 | 3.3, 4.4 %};`
		`print_matrixf($$m, 2, 2);`
		`/* { 1.1, 2.2`
		`3.3, 4.4 } */`

print_matrixt(int(%3%)[[]] m, int len, int wid)	Prints matrix of tuples, pass in pointer to tuple matrix, length, and width as arguments	`m = {% (%1,1,1$), (%2,2,2%)`
		`|(%3,3,3%), (%4,4,4%) %};`
		`print_matrixt($$m, 2, 2);`
		`/* { (1,1,1), (2,2,2)`
		`(3,3,3), (4,4,4) } */`

add_rowi(int[[]] oldm, int[[]] newm_ptr, int[] row, int len, int wid)	Adds an int row to a int matrix and returns a pointer to a new int matrix, pass in pointer to old int matrix, pointer to new int matrix, length of old matrix, and width of old matrix as arguments	`int[2][2] m1;`
		`int[3][2] m2;`
		`int[[]] newm;`
		`int[2] r;`
		`m1 = {% 1, 2 | 3, 4 %};`
		`r = [5, 6]`
		`newm = add_rowi($$m1, $$m2, $r,`
		`m1.length, m1.width);`
		`/* { 1, 2`
		`3, 4`
		`5, 6 } */`

add_rowf(float[[]] oldm, float[[]] newm_ptr, float[] row, int len, int wid)	Adds an float row to a float matrix and returns a pointer to a new float matrix, pass in pointer to old float matrix, pointer to new float matrix, length of old matrix, and width of old matrix as arguments	`float[2][2] m1;`
		`float[3][2] m2;`
		`float[[]] newm;`
		`float[2] r;`
		`m1 = {% 1.1, 2.2 | 3.3, 4.4 %};`
		`r = [5.5, 6.6]`
		`newm = add_rowf($$m1, $$m2, $r,`
		`m1.length, m1.width);`
		`/* { 1.1, 2.2`
		`3.3, 4.4`
		`5.5, 6.6 } */`

e. Syntax

e1. Declaration and Initialization

Variable Declaration and Initialization

All variables must be declared before use. A declaration specifies the variable type and the variable name. After the variable is declared, it can then be initialized after. A variable cannot be named "_i" because this is the built-in index variable for the matrix for loop.

General Examples

```
variable_type variable_name;
variable_name = literal;
```

Specific Examples

```
int x;
x = 6;
float y;
y = 3.7;
```

Tuple Declaration and Initialization

All elements in a tuple must be of type int, and tuple length must be 3. A proper declaration specifies the int type, tuple length, and tuple name. A tuple initialization starts and ends with " (% " and " %) " respectively.

Examples

```
int(%tuple_length%) tuple_name;

int(%3%) tup;

tup = (% 1, 2, 3 %);
```

Row Declaration and Initialization

All elements in a row must be of the same type, and elements in a row can only be ints, floats, or tuple of ints. A proper declaration specifies the element type, row length, and row name. A row initialization starts and ends with " [" and "] " respectively.

Examples

```
type[row_length] row_name;

int[6] r;
r = [1, 2, 3, 4];

int(%3%)[2] tuple_row;
tuple_row = [(% 2, 1, 5 %), (% 6, 0, 2 %)];
```

Matrix Declaration and Initialization

All elements in a matrix must be of the same type, and elements in a matrix can only be ints, floats, or tuples of ints. A proper declaration specifies the element type, the row length, column length, and the matrix name. A matrix initialization starts and ends with " {% " and " %} " respectively. Bars, " | ", separate each row so that there are tokens.

Examples

```
type[row_length][col_length] matrix_name;

float[3][3] m;
m = {% 1.1, 2.1, 3.1 | 4.0, 5.1, 6.2 | 7.4, 8.5, 9.9 %};

int(%3%)[3][2] tuple_m;
tuple_m = {% (% 1, 2, 1 %), (% 3, 4, 3 %) | (% 5, 6, 5 %), (% 7, 8 ,7
%) | (% 9,10, 9 %), (% 11, 12, 11 %) %};
```

The following are the matrices that m and tuple_m represent.

```
m:
{ 1.100000, 2.100000, 3.100000
  4.000000, 5.100000, 6.200000
  7.400000, 8.500000, 9.900000 }

tuple_m:
{ ( 1, 2, 1 ), ( 3, 4, 3 )
  ( 5, 6, 5 ), ( 7, 8, 7 )
  ( 9, 10, 9 ), ( 11, 12, 11 ) }
```

Pointer Declaration and Initialization

Pointers are used to access elements in a matrix. There are two main pointers that have been created: row pointers and matrix pointers. Within each of these categories, the pointers can deal with ints, floats, and tuples. Pointers can also be referenced, dereferenced, and incremented.

Declare a row pointer:

```
int[] x;
```

Declare a matrix pointer:

```
int[[]] x;
```

Increment a matrix or row pointer:

```
~~x;
```

Dereference a matrix or row pointer:

```
#x;
```

Examples

Row Pointers:

```
int[4] x;
int[] y; /* creation of a row pointer */
int q;

x[0] = 9;
y = $x; /* pointing pointer to row reference */
q = #y; /* dereference pointer to get the first value in row x
*/
```

Matrix Pointers:

```
int[2][2] y;
int[[]] p; /* creation of matrix pointer */
int q;

y[0][0] = 1;
y[0][1] = 2;
y[1][0] = 3;
y[1][1] = 4;

p = $$y; /* pointing pointer to matrix reference */
p = ~~p; /* increment pointer */
p = ~~p;
q = #p; /* dereference pointer to get the value, should get 3
from this*/
```

e2. Expressions

Arithmetic and Matrix Operations

Assignment operators are binary operators with right-to-left associativity. Arithmetic and matrix expressions are mathematical operations with left-to-right associativity. In addition to arithmetic operations with ints and floats, arithmetic operations can be done on matrices of type int and floats and tuples as well. The following are the four additional types of arithmetic operations allowed: matrix and matrix operations, matrix and scalar operations, tuple and tuple operations, and tuple and scalar operations. Operations with a datatype and a scalar include addition, subtraction, multiplication, and division. Operations with two of the same datatypes include addition and subtraction. All operations can only be done with variables of the same type.

Examples

Tuple subtraction:

```
int(%3%) t1;
int(%3%) t2;
int(%3%) t3;

t1 = (% 1, 2, 3 %);
t2 = (% 1, 2, 3 %);
t3 = t1 - t2; /* (% 0, 0, 0 %) */
```

Matrix addition:

```
int[2][2] m1;
int[2][2] m2;
int[2][2] m3;

m1 = {% 2, 4 | 6, 8 %};
m2 = {% 1, 3 | 5, 9 %};
m3 = m1 + m2; /* {% 3, 7 | 11, 17 %} */
```

Scalar multiplication:

```
float[2][2] m1;
float[2][2] m2;
m1 = {% 1.0, 2.0 | 3.0, 4.0 %};
```

```
m2 = 5.0 * m1;   /* {% 5.0, 10.0 | 15.0, 20.0 %} */
```

Scalar division:

```
int(%3%) t1;
int(%3%) t2;

t1 = (% 2, 4, 6 %);
t2 = t1 / 2; /* (% 1, 2, 3 %) */
```

f. Control Flow

f1. Statements and Blocks

Each statement is followed by a semicolon.

Example:

```
int x = 6;
```

Blocks are surrounded by brackets.

Example:

```
if (x == 6) {
    print("x is 6");
} else {
    print("x is not 6");
}
```

f2. If-Else If-Else

if-else if-else statements are blocks. When the "if" condition is not met, the program will check for any other conditions, which is specified by "else". "else if" is required if checking for more than two conditions; "else" will suffice otherwise. An "if" can exist without a corresponding else, but an "else" cannot exist without a corresponding "if". When a condition is met, the program will execute the code in the corresponding block, and ignore all subsequent "else" conditions. An "if" statement requires a condition, while an "else" statement does not.

Example:

```
if (x > 6) {
    print("x is greater than 6");
```

```
} else {
    print("x is less than or equal to 6");
}
```

f3. Loops

For Loop

The J-STEM for loop operates like Java's. There are 3 fields to the condition of a for loop. First, an index variable is initialized to some value, then the stop condition is specified, and then the increment/decrement of the index variable is given (using operators += or -=). The block is looped through according to these fields. The index variable must be declared before the for loop.

```
for ( index; stop_condition; step_value) {
     ...
}
```

Matrix For Loop

An enhanced version of the standard for loop used to iterate easily through matrices. The enhanced for loop uses the keyword "in" to iterate through a pre-declared and initialized matrix, returning the value of the next row on each iteration. row is NOT declared or initialized earlier in the program.

```
for ( row in matrix ) {
     ...
}
```

While Loop

The while loop also operates like Java's. In the condition, there is only one field, for specifying the stop condition. Usually, this stop condition utilizes operators such as ==, <, >, etc.

```
while ( stop_condition ) {
     ...
}
```

Examples:

```
int i;
```

```
for (i = 0; i < 10; i += 1) {
     ...
}

float[2][2] m1;
m1 = {% 1.0, 2.0 | 3.0, 4.0 %}
for (row in m1) {
     print_rowf(&row, m1.width);
}
// [ 1.0, 2.0 ][ 3.0, 4.0 ]

int i;
i = 0;
while (i < 10) {
     ...
     i = i + 1;
}
```

g. Functions and Program Structure

g1. Functions and Function Calls

To declare functions, use keyword ‘def’.If there is no return value for the function, use keyword ‘void’ and ‘main’ followed by parenthesis and brackets for the function body. If the function returns a value, state the type of the return of the function, function name, and argument type and names in parenthesis followed by brackets to enclose the function body. Functions have to be defined as such and implemented before being called. The main function must return 0.

Function Declaration

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

def  return_type function_name( arg_type arg_name ) {
     return expression;
```

```
}
```

Example:

```
def int[2][3] add(int[2][3] a, int[2][3] b) {
     int[2][3] new_matrix;
     new_matrix = a + b;
     return new_matrix;
}
```

g2. Scope Rules

The scope of a variable depends on when it is declared within a function. If the variable is declared at the outermost level of the function (i.e. at the beginning of the function), then it is accessible throughout the function. A variable cannot be declared at the beginning of a loop, only at the beginning of a function (see example 2). If a previously declared variable is initialized in the condition of a loop and then modified inside the loop, then the modified version is accessible outside that loop (see example 3).

Example 1

```
def void main() {
    int x = 5;
    print(x); /* will print 5 */
}
```

Example 2

```
def void main() {
    int i;
    i = 0;
    while(i < 5) {
     int square; /* error */
     square = i * i;
     print(square);
     i = i + 1;
    }
}
```

Example 3

```
def int main() {
     int i;
     for (i = 0; i < 4; i=i+1) {
          print(i); # prints 0 1 2 3
     }
     print(i); # prints 4
     return 0;
}
```

g3. Block Structure

Functions, conditionals, and loops will all be written using block structure, defined by an opening bracket "{" and a closing bracket "}". Variables declared within brackets cannot be accessible outside the block.

Example Code

This is example code in our language that prints the sum of every value in an int matrix. It highlights blocks and nested blocks.

```
def void print_sum(int[[]] m, int len, int wid) {
     int i;
     int sum;
     int elements;
     sum = 0;
     elements = len * wid;
     for (i = 0; i < elements; i=i+1) {
          sum = sum + #m;
          m = ~~m;
     }
     print(sum);
}

def int main() {
     int[2][2] m;
     m = {% 1, 2 | 3, 4 %};
     print_sum($$m, m.length, m.width);
     return 0;
}
```

4. Project Plan

a. Planning Process

The JSTEM team assigned project roles and set bi-weekly times to meet. Many times, the team members worked remotely and the team typically met more than twice a week. The JSTEM team also met with their TA, Alexandra Medway, every week who offered advice and helped ensure that the team stay on track in terms of progress.

At first, the team worked on the parser, scanner, ast, and preliminary codegen through Google Docs so that all the team members could participate in modifying the initial skeleton of the project. In general, the team employed GitHub, Google Docs, and messaging services to work together and to communicate.

b. Specification

When determining how JSTEM would look, the team drew on inspiration from Java and Python. For instance, for loops were modeled after for loops in java. Declaring functions with the word "def" was inspired by Python. An initial language reference manual (LRM) was created for JSTEM, but was modified and adjusted as the language was developed.

c. Development and Testing

The team was advised to complete the scanner, parser, and AST quickly because the code generator and semantic analysis would take the most time. A framework of the compiler was first built from end-to-end to simply get "Hello, World" to print out. From there, each feature was integrated one-by-one from end-to-end in the same way that "Hello, World" was developed. The initial test suite was also modeled off of microc. However, as more features were added, not all of them were included in the microc-esque tests. The developers then added in additional tests to ensure that the new features they were including worked. The team worked together through the initial parser, scanner, and AST together as a group. After "Hello, World" worked, every member chose a feature to work on end-to-end.

d. Style Guide

- We wrote our compiler in OCaml, and adhered to established OCaml programming practices in order to to make our code consistent and clear.
- We wrote our JSTEM program code using the following guidelines:
    - File names end in the extension .JSTEM
    - A main function must always be included

- For indentations, we used a tab that was equivalent to 4 spaces
- Ident statements that come in between curly braces (i.e. code within a function, code within a conditional, or code within a loop)
- Both variable and function identifiers begin with lowercase letters and are camelcase
- Block comments are indented at the same level as the surrounding code
- Only have one statement per line, each statement is followed by line break
- A line should not exceed 80 characters

e. Timeline

Date	*Milestone*
Feb 8	Proposal
Feb 22	LRM
Mar 21	Initial Scanner & Parser
Mar 23	AST, Pretty Printing
Mar 26	Initial Test Suite, Codegen
Mar 27	Hello World
April 25	Matrix Handling
May 6	Semantic Analysis
May 7	Pointers
May 8	Bug Fixes and Branch Merges
May 10	Final Report

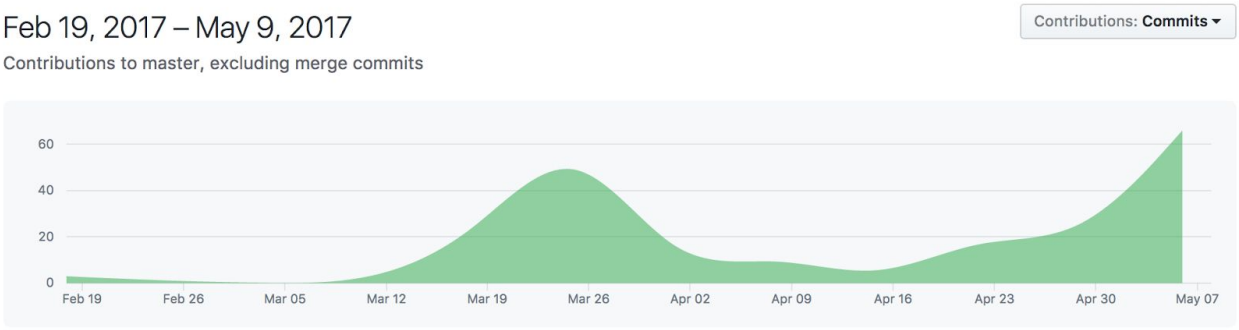
f. Team Roles

Julia Troxell - Manager
Samantha Stultz - System Architect
Tessa Hurr - Language Guru
Emily Song - Language Guru
Michelle Lu - Tester

g. Software Development Environment

Operating Systems: Mac OS Systems, Ubuntu 15.10 on VirtualBox
Languages: OCaml (OPAM to install), Java & Python for inspiration
Text Editor: Sublime, Vim
Version Control: Git, GitHub
Documentation: Google Docs
Continuous Integration: Travis CI

h. Project Log



The log above shows the commit timeline. There is a large peak in the middle towards the end of March. This is when we did a lot of the work on the scanner, parser, ast, and codegen to get "Hello, World" working. After this time, we were committing code to master more consistently.

Commit History:

```
cb69ba523b7c812295c0d59595d839528d03b967 Update README.md
907e8b465dc395c827e5ef2a29de151a00cb2b80 code cleanup
b0f84c213c1f0e2b22dfd1ba82b760da39faafed added both demos
97942a1c9543c49f4b85081594cf1b4308a00ae7 demo
0f7ea640eba4944fe267545f59f133f37cd7a8fc Merge branch 'master' of
https://github.com/MichelleLu1995/PLT
164d74bc8d57899a137f7a175c7af814f00d08e4 demo1 using the MForLoop
e7eeae95513874a2d2e56007508b0022a677b71a Merge branch 'master' of
https://github.com/MichelleLu1995/PLT
6850a753b0fee311892e7b90fec7b0a968edffcf removing some pointer stuff
5cc6be6825d378f802cb1d763b5fb00e2dc4e3db Merge branch 'master' of
https://github.com/MichelleLu1995/PLT
53403b912f68eb22c663f7c49d8a2202d96e46f4 added to demo
15f904380efcf46812eb222664f9d1a0c9da4d0d Merge branch 'master' of
https://github.com/MichelleLu1995/PLT
428b0c188837c5ceb9bd89cc23a7ae7fa1131041 fix tests and printing
29a5e8d46d16a1a72f54770f41def372274e4b8f Merge branch 'master' of
https://github.com/MichelleLu1995/PLT
2a6aeb96a8669fe292009a0b2c56e6a3911eb438 added demo
fdb0ec98bc04530b824bf23c459eba27e6de05ee adding pointer tests to the test suite
```

```
af8384c6cab9c1dfc540e656156415eab5d655f2 tests should work except for files
2a1e2c301b25740328e718d801f0741bb5bbd4ca Merge branch 'master' of
https://github.com/MichelleLu1995/PLT
aa74ff560e51feecfce448fcf5136de093bb23de Update codegen.ml
f627cc64db02cdea0e12e776ac0f80a6ab3b777b more tests
7cfa3cf0d2b54b0a8aa30a871f88f1b87f23e1a3 resolve conflicts
7c26cbe3a5512cd1cfe56e69003cc803b3b4a1cc changed test outputs so no new line on
print
a9f9e48675d6d6448e82e01e9c40c8dfb7fca11e stdlib function to append a row to a
matrix
a765ad47606260afc407437a6f61eea058b9112b merge conflicts fixed
a2e3cc69c41a4cc094ed6029d485fb6103e6b94f fixed merge conflicts
7b305faa150b0cf2bd966d9a65f2479739834a4c merge conflict fixes
dee1fe175ade61ba370fd45f104a67b9bb6c2417 more string stuff
bf2f380a80c23e759b401eaad62554800300f9ea write to file
78c34d279e3097f420b8a9b555277ef68930def0 changed test to print out
tuple/row/matrix with std lib func
02ebd43f9532c374585d5cb8bebf9ae60cf69c70 removed native file
dad3b5176e201b45c18659890061b60167c4860e Merge pull request #47 from
MichelleLu1995/arith-tests
75337f223776c6e18076ec7d03fc200e8121115c all tests pass
b72764aff9e45707147faaaab1ab74d767c432ab Merge branch 'master' of
https://github.com/MichelleLu1995/PLT
b30c1376fcc496e99c275d24d06b1c707aadf88b Merge pull request #46 from
MichelleLu1995/arith-tests
7ced60549a98c49e6b681bb955fbf4cf1fe01a91 fixed last of the tests
7d4e83810559cc8c0171a5898ce453fc74418e34 Merge branch 'master' of
https://github.com/MichelleLu1995/PLT
d32c176d3e992322e01b57c9656add3761ac89b9 pattern matching to remove warnings
when make
611e3a19422a00832ba5d9d0194e41fa9f9b2f8f Merge pull request #45 from
MichelleLu1995/arith-tests
ab0a96744cc0521b8222927f95fa4861b85beefc deleted old testing error files, fixed
another fail test
1cad7664ec0117a2146556e1b73bdbfba48c6e9e Merge pull request #44 from
MichelleLu1995/arith-tests
96a23dfb9f93080a109255157f5b8a510b1168d8 added bool operators
a0a44f9dd2d257403d35b874b4c7b2c6400ef5cc added string return type
41f08581c594d1e7a155f33994e2f158e8dce108 minor fixes to arith tests
fb37c3bb59e7d2b9f8e30301233cb04efe51beda tests for files delim, (str)len, and
str to int
f6a52ee3dde1b4efd3466bc1e7d8493e0fed622d Merge branch 'master' of
https://github.com/MichelleLu1995/PLT
374f00429731ddf53ad7a230f25ef93279a9884d tests for files and delimiters and
string to int
a68538e7d27db588c74fe5673421fa97adb407ff fixed more scalar tuple tests
71878427291de65aa8c86e1e053003b6902a1b92 fixed tuple scalar tests
303dded3a00ef50ca887087d68e9ff26b99925aa Update README.md
bbc44754d62e14a9e9be85997ae90c587ba374df merged conflicts fixed
0d066ae2002d40f521217079526201fb614a00f2 Merge branch 'MForLoop'
```

```
eb7eea5ccd9b2acd8aca48fa2b12aa25347a8736 Merge pull request #43 from
MichelleLu1995/fix-tuple-tests
c120f445e55624b78d445814aa0a27d4e9fbc0c5 readfile
051af380e9f0df6a6d77b9633ea728681aa1b265 fixed compiling issues
5fb692b427897dccef790329644d6ed7f0136d81 Merge branch 'master' of
https://github.com/MichelleLu1995/PLT
26f5735430b3546f5c2185af8af7cd96ff1c7ae0 merge conflicts fixed
689f1a5413626673a224974957444c4d553b0660 edited tests for only tuples of length
3
2694d543625ff24a42de3a98687ed026440a797e Merge pull request #42 from
MichelleLu1995/files3
4288cc408a6433a9a0c587df4ca88d92c084557b removed unused tokens
654ea7d98c1f44f049a57e0fe8bf52c9e55e31c3 fixed splitstr
e0a72ce15e57a9da8d0720da079c8ad255862887 stdlib printing success tests
82bcc37f17c8902f480bce489610d7c4a5fea1c6 fixed merge conflicts
3f43af12eacf3d1dce53d9f7062c6fb44d3277bd fixed fib test
1210522e8a40c4fb3b8a4fd5cf5498c4741ae0d8 std library functions for printing all
tuples/rows/matrices
18822716e41f345203689cc5eedfa9ca4e7a2119 added matrix/tup scalar mult and div
with tests
aa94a892321bec662c6e0b82cc361080379a3203 segmentation fault issues
6ec943fba44f96f046cabcf416cacfd4b6aac5b6 merge conflicts
3c8a6183e45c0a3f0228320578ad7b3f039bfdda matrix and tuple add/sub with scalar
working and tests
dae736016cd55d3ec7d0df732ccb49d2a930ce40 jk did it wrong but now they are all
resolved
f65eee5f48453a8ebdce8d943898de3844cb22d9 fixing freaking merge conflicts
188be4d5fa37b012c3e92648e8f8a0151332ecfe commit before merge
6c79ff785119afcd2473177e66e6bf171d8f2473 fixed code for more complex arithmetic
4c7544ca2a541a522b1f112925b635f4642c5cb6 added back mult and div into the
semant
d01e0bd891b88dfc26012dbc0323d78738bc8acf fixed add2 test
462b111ac1a2c64cca62227b69f1a269452a8dfd fixed add/sub tuple tests
a00f9dfd359bf5791c6fd6765ae9008b28340535 Merge branch 'fix-tuple-tests' of
https://github.com/MichelleLu1995/PLT into fix-tuple-tests
76036e0aacb281818484358ecf7d5e287277d5e1 conflicts fixed
d3157580dd3a8ee86bcb76212ff61a6e1bfc05e1 checking for operands improved
728c84b0902192db6f9d7b29c3e8f005aab273cc fixed add1 and add2 tests
61b8d27d8ffd729b2cff1ab0294a8266d2855e2f fixed matrix sub floats test
6ab956cff979db897c85913a647cfae0e63eb070 fixed semant warnings
97a5ca1f158857ad36eed76ecee424f38cf0649e fixed matrix floats test
bc366362d7a5fb336e82f7de16943df562a9404b fixed most of tuple/matrix codegen
issues
5c04df4ecb872ae95cc66c9faca70e4e85269422 semant analyzer for tuple/matrix ops
1ff68c8dfeb930afec5c9dbbcc2f62ae54737f08 removed automatic newlines from
printing, added function matrix.width
e792f8f0b7bccefab19449ff77351c9e88493237 Merge branch 'master' of
https://github.com/MichelleLu1995/PLT
0ec12de66a279459597e847ffc5b4ef3552c61e4 Revert "Revert "merged
addingmatrixacces branch with matrix/row of tuple types""
```

```
c12301c2d8124f0aae5ec0cd6fdb698f08ab15b1 Revert "merged addingmatrixacces
branch with matrix/row of tuple types"
35b9bae5cb124c3491400fe4e6388fc90d796404 Merge branch 'master' into pointer2
7fe82c092592a0b2b71ffe2c735627caeb39a1ca Fixed shift reduce on matrix pointer
aabe288aa928cfe023703f218a19da3c6fa63e70 merged addingmatrixacces branch with
matrix/row of tuple types
182870c1b188e08a9e932e5544f0d368df283f05 function to return the type of a row
or matrix
308971f15f1cafe8112e45b9c9ef499c3c5c88a0 fixing more merge conflicts
0e425b6f0b2fc7bcfe083ccbc3ae8b391e0da387 fixing something
dcdb7d6f819fbd73252419d98a7db55e4d69fe6d merge
8175c739eceb78df77c005407f49d9c4fcc721c6 Merge pull request #39 from
MichelleLu1995/MForLoop
9ca2914ed615ebfa046a179c5b413a92093c4aab Merge branch 'master' into MForLoop
6fc8fb9e63359b97a53ffe5bf72091e56c947166 added tests for matrix and row of
tuple type working
3a2d3c5a8fbd8fbbba6d5bb998117fcb56b96d8e Merge pull request #38 from
MichelleLu1995/file-reading
05f01eb28cd04063762788fdd692a0ec4475a230 needs to go into master
03bd54fdd1002b86c5fc31cc8a2120fcb42f15d4 matrix.length and row.length working!
6081e07285e9ae17985e327077dd357de3a342a6 matrix.length and row.length working!
fd61c11d379abfa46ab015a79b0596ab3b62627a Merge pull request #37 from
MichelleLu1995/pointer2
d3746a3f9bfb7b3f93a6bb73ce2e3e088b934ab3 Merge branch 'master' into pointer2
f6d3bc87776bf6e5848dd0ec1cf716c3fd147a54 Merge pull request #36 from
MichelleLu1995/MForLoop
05ce045563edcba52eedd6b9ee9dde6028f0718e Merge branch 'master' into MForLoop
06a240f698c08a5fbe7c8bca46135c58c33bb312 matrix pointer working
cab42cefe6eb82112d1aa76bdf7be0d5f6d33498 MForLoop works with merged Binop code
9bfccb4389bfdb4b9711617b0faf54409cc65206 Merge pull request #35 from
MichelleLu1995/codegen-warnings
a412aaf258a4d5a0e7eeb88fb29a3c7625d7b357 fixed first half of codegen warnings
5a6387fbac0f41758655561d96ef833be06b0729 fixed first half of codegen warnings
20d29cfad5fe66a6ff9f3dde9a6626bdad28d2b3 file was in scanner apparently
e1623e290e92242c3c81ebe68f2a780b1d5d0c96 hopefull fix conflicts
206df1f59f1e2ab556d428bf1871ab0cdf8b0176 fixing merge conflicts
dd69195d5a5f208c9f203222bf426badf282b97f now compiles
84f9487f20f3020fae56774f1aa8e9b036a27433 Revert "merge fixes" fixing mfor and
compiling
fab8860a51fe82d4cb58da1ddb1c51c0e588c5e2 conflict fix
fd2a985602e522244b6259001eed0b250962cd00 fix merge
4b0cf3420b8ceef900280bd1cac1b6d7da265bb3 remove unecessary files
d704a8f319f82042b63087d09def7c9d650491fd Revert "fix tokenize"
7446e435e0b323e402a38d49f02a6b36d8b541aa merge fixes
db86aff2d28865dc76d66aeb854f91fba2d67ec4 now you can use any word, not just row
in 'for row in matrix'
7d00e0ec578f2aa56cf41fa1e6af23f253e341a7 Merge pull request #34 from
MichelleLu1995/MForLoop
fde6db7ba66b008dfa45798c1a7be68e0b627d8c Merge branch 'master' into MForLoop
```

```
519d28d4e0caa0ecb5cddcddc41fc5efde309056 MForLoop works for int/float matrices,
returns each row - not tested for tuples
01bc62bce9ca961130b32eaf3c8e47ed0040f98e merge fix
9e1284978b5d1eae8f1431db1beb1d0903fd01c2 merge conflicts
3aa35b0183fd94035bf275a8ffedcf9827beb374 added more tests and working on
matrix/row of tuple type
0beb079386defa4efb8f52354a39fa69b69180f4 fix tokenize
48be6a5ccd9e717267b21e661df23c5c48eaadbd Merge pull request #33 from
MichelleLu1995/matrix-ops
6cb027230497e96e2e38e7c5a36bffc262a02b6e Merge branch 'master' into matrix-ops
d706db3010f3b41cd6f0e2ec7f2ee45c375da9f2 pointer dereferencing works
e18ea729f5a1a8ffc652a19d3ec5f13787fe67d6 adding in pointer incrementing
8445c741bde634a68194161a28255c70fa14e778 merge fixes
8999d1c3c07516c68c0b279511f828808b802d62 arrays update
bae9c6569279bbbd96ff946432c3a41729b1844c Merge pull request #32 from
MichelleLu1995/pointer2
4b1fcbbf1cbadc634a4d72933ca74e681608b126 Merge branch 'master' into pointer2
37881aa07c64f72b31c624939314f102b6eadfb6 fixed adding two ids
58b3c2c6d36d0a116ffab4fb55eabd1c05c64ce5 added code + tests for matrix
subtraction
20d1b12d53510414a5a3d64e8f774be63a52994f deleted row ops tests
bdb5af4257079e97f2f6592fd0bf1ddb0842dd86 add matrix float adding code +
renamed/added tests
36f3f964adef5dea46d953784567602f0b4e3b53 adding matrices of intsgit add
codegen.ml semant.ml
53808caa269096070e126018662b22bd7b1bdfc4 can now subtract integer tuples of
variable length
2127045581cf176d870ec521c44ffdc4b8e9f7ba changed one of the tuple adding tests
72d10ba5851578663a3e5ffed26d9356edf4f91d can now add integer tuples of variable
lengthsgit add codegen.ml semant.ml tests/compiler_tests/test-addtuple.JSTEM
591bd7d6557d2de8ca7fbdc917e6b6c93b54d4e2 pointer successls
dd98ef65f073c285c0177e918916c6abade0f331 tuple code hard coded, but now works
9cc5c9a5d3449542305e567f85dab099cf56449e code cleanup
20a4504b2bb3aaa61f1aa4adefa442c29834b054 somehow fixed it
33e5b0759bf81439e941f4d6ebb9021fef723532 remove bad files
c648e8a51b8ed529f31f1cc7c6361eae9a6ee9c9 Matrix for loop
14795c2a1ad8e7d424144ad35e94b11366fc31ae matrix for loop
490c94d0fe7758033fd775c2b19f7a8d9441f072 can add int tuples of length 3, but
for loop not working
279573347c7b0a6b29b51fafa1368aa1f66732ee broke ints and floats but adding
tuples
4a5d5cce64603ee5f346fa3310381572664df090 having issues. don't pull
b6cc3f2475c2af05b49f74a0a141fdc8b29b07bd beginning to add vector int ops
43a48de05807274cd959459a1177d419c78b922f minor fix to tuples test
:
20d1b12d53510414a5a3d64e8f774be63a52994f deleted row ops tests
bdb5af4257079e97f2f6592fd0bf1ddb0842dd86 add matrix float adding code +
renamed/added tests
36f3f964adef5dea46d953784567602f0b4e3b53 adding matrices of intsgit add
codegen.ml semant.ml
```

```
53808caa269096070e126018662b22bd7b1bdfc4 can now subtract integer tuples of
variable length
2127045581cf176d870ec521c44ffdc4b8e9f7ba changed one of the tuple adding tests
72d10ba5851578663a3e5ffed26d9356edf4f91d can now add integer tuples of variable
lengthsgit add codegen.ml semant.ml tests/compiler_tests/test-addtuple.JSTEM
591bd7d6557d2de8ca7fbdc917e6b6c93b54d4e2 pointer successls
dd98ef65f073c285c0177e918916c6abade0f331 tuple code hard coded, but now works
9cc5c9a5d3449542305e567f85dab099cf56449e code cleanup
20a4504b2bb3aaa61f1aa4adefa442c29834b054 somehow fixed it
33e5b0759bf81439e941f4d6ebb9021fef723532 remove bad files
c648e8a51b8ed529f31f1cc7c6361eae9a6ee9c9 Matrix for loop
14795c2a1ad8e7d424144ad35e94b11366fc31ae matrix for loop
490c94d0fe7758033fd775c2b19f7a8d9441f072 can add int tuples of length 3, but
for loop not working
279573347c7b0a6b29b51fafa1368aa1f66732ee broke ints and floats but adding
tuples
4a5d5cce64603ee5f346fa3310381572664df090 having issues. don't pull
b6cc3f2475c2af05b49f74a0a141fdc8b29b07bd beginning to add vector int ops
43a48de05807274cd959459a1177d419c78b922f minor fix to tuples test
61ab00bcba27bdc70ccf275d6bc462a96c9a1ce6 added tests for adding tuples, rows,
and matrices
:
20d1b12d53510414a5a3d64e8f774be63a52994f deleted row ops tests
bdb5af4257079e97f2f6592fd0bf1ddb0842dd86 add matrix float adding code +
renamed/added tests
36f3f964adef5dea46d953784567602f0b4e3b53 adding matrices of intsgit add
codegen.ml semant.ml
53808caa269096070e126018662b22bd7b1bdfc4 can now subtract integer tuples of
variable length
2127045581cf176d870ec521c44ffdc4b8e9f7ba changed one of the tuple adding tests
72d10ba5851578663a3e5ffed26d9356edf4f91d can now add integer tuples of variable
lengthsgit add codegen.ml semant.ml tests/compiler_tests/test-addtuple.JSTEM
591bd7d6557d2de8ca7fbdc917e6b6c93b54d4e2 pointer successls
dd98ef65f073c285c0177e918916c6abade0f331 tuple code hard coded, but now works
9cc5c9a5d3449542305e567f85dab099cf56449e code cleanup
20a4504b2bb3aaa61f1aa4adefa442c29834b054 somehow fixed it
33e5b0759bf81439e941f4d6ebb9021fef723532 remove bad files
c648e8a51b8ed529f31f1cc7c6361eae9a6ee9c9 Matrix for loop
14795c2a1ad8e7d424144ad35e94b11366fc31ae matrix for loop
490c94d0fe7758033fd775c2b19f7a8d9441f072 can add int tuples of length 3, but
for loop not working
279573347c7b0a6b29b51fafa1368aa1f66732ee broke ints and floats but adding
tuples
4a5d5cce64603ee5f346fa3310381572664df090 having issues. don't pull
b6cc3f2475c2af05b49f74a0a141fdc8b29b07bd beginning to add vector int ops
43a48de05807274cd959459a1177d419c78b922f minor fix to tuples test
61ab00bcba27bdc70ccf275d6bc462a96c9a1ce6 added tests for adding tuples, rows,
and matrices
4b5168c71e375324e997503fdb457223cb7e6618 merge file-reading to addingmatrix
```

```
d4843e900cc8498a0d61b0b7d0221ba87c454ea3 Revert "Merge branch
'AddingMatrixAccess' into master"
25ab042fdb7201841cf1bfb91054eada22831e57 Merge branch 'AddingMatrixAccess' into
master
:
20d1b12d53510414a5a3d64e8f774be63a52994f deleted row ops tests
bdb5af4257079e97f2f6592fd0bf1ddb0842dd86 add matrix float adding code +
renamed/added tests
36f3f964adef5dea46d953784567602f0b4e3b53 adding matrices of intsgit add
codegen.ml semant.ml
53808caa269096070e126018662b22bd7b1bdfc4 can now subtract integer tuples of
variable length
2127045581cf176d870ec521c44ffdc4b8e9f7ba changed one of the tuple adding tests
72d10ba5851578663a3e5ffed26d9356edf4f91d can now add integer tuples of variable
lengthsgit add codegen.ml semant.ml tests/compiler_tests/test-addtuple.JSTEM
591bd7d6557d2de8ca7fbdc917e6b6c93b54d4e2 pointer successls
dd98ef65f073c285c0177e918916c6abade0f331 tuple code hard coded, but now works
9cc5c9a5d3449542305e567f85dab099cf56449e code cleanup
20a4504b2bb3aaa61f1aa4adefa442c29834b054 somehow fixed it
33e5b0759bf81439e941f4d6ebb9021fef723532 remove bad files
c648e8a51b8ed529f31f1cc7c6361eae9a6ee9c9 Matrix for loop
14795c2a1ad8e7d424144ad35e94b11366fc31ae matrix for loop
490c94d0fe7758033fd775c2b19f7a8d9441f072 can add int tuples of length 3, but
for loop not working
279573347c7b0a6b29b51fafa1368aa1f66732ee broke ints and floats but adding
tuples
4a5d5cce64603ee5f346fa3310381572664df090 having issues. don't pull
b6cc3f2475c2af05b49f74a0a141fdc8b29b07bd beginning to add vector int ops
43a48de05807274cd959459a1177d419c78b922f minor fix to tuples test
61ab00bcba27bdc70ccf275d6bc462a96c9a1ce6 added tests for adding tuples, rows,
and matrices
4b5168c71e375324e997503fdb457223cb7e6618 merge file-reading to addingmatrix
d4843e900cc8498a0d61b0b7d0221ba87c454ea3 Revert "Merge branch
'AddingMatrixAccess' into master"
25ab042fdb7201841cf1bfb91054eada22831e57 Merge branch 'AddingMatrixAccess' into
master
1ab8f694a40f81c0bb4747980d7c44f1f42948dc added tests to check semantic and
added tuple type to row and matrix in codegen
9fcf939fcf7181c1f9ef66d6c7a058c14f19daea Merge pull request #30 from
MichelleLu1995/file-reading
056cab3f6c398e8660059a626bdf689268f538db working files, but weird segmentation
fault on longer files
56b32d911bef9bd3f20ecf161b30a5a8553ba1a5 might work, but need to add file type
c3832e6e81d1bb9c78d3cacd5c276a5e77dfa9fe resolved merge
effa17505556d14eaab405448246a10b4840f61b added to sematic
:
20d1b12d53510414a5a3d64e8f774be63a52994f deleted row ops tests
bdb5af4257079e97f2f6592fd0bf1ddb0842dd86 add matrix float adding code +
renamed/added tests
```

```
36f3f964adef5dea46d953784567602f0b4e3b53 adding matrices of intsgit add
codegen.ml semant.ml
53808caa269096070e126018662b22bd7b1bdfc4 can now subtract integer tuples of
variable length
2127045581cf176d870ec521c44ffdc4b8e9f7ba changed one of the tuple adding tests
72d10ba5851578663a3e5ffed26d9356edf4f91d can now add integer tuples of variable
lengthsgit add codegen.ml semant.ml tests/compiler_tests/test-addtuple.JSTEM
591bd7d6557d2de8ca7fbdc917e6b6c93b54d4e2 pointer successls
dd98ef65f073c285c0177e918916c6abade0f331 tuple code hard coded, but now works
9cc5c9a5d3449542305e567f85dab099cf56449e code cleanup
20a4504b2bb3aaa61f1aa4adefa442c29834b054 somehow fixed it
33e5b0759bf81439e941f4d6ebb9021fef723532 remove bad files
c648e8a51b8ed529f31f1cc7c6361eae9a6ee9c9 Matrix for loop
14795c2a1ad8e7d424144ad35e94b11366fc31ae matrix for loop
490c94d0fe7758033fd775c2b19f7a8d9441f072 can add int tuples of length 3, but
for loop not working
279573347c7b0a6b29b51fafa1368aa1f66732ee broke ints and floats but adding
tuples
4a5d5cce64603ee5f346fa3310381572664df090 having issues. don't pull
b6cc3f2475c2af05b49f74a0a141fdc8b29b07bd beginning to add vector int ops
43a48de05807274cd959459a1177d419c78b922f minor fix to tuples test
61ab00bcba27bdc70ccf275d6bc462a96c9a1ce6 added tests for adding tuples, rows,
and matrices
4b5168c71e375324e997503fdb457223cb7e6618 merge file-reading to addingmatrix
d4843e900cc8498a0d61b0b7d0221ba87c454ea3 Revert "Merge branch
'AddingMatrixAccess' into master"
25ab042fdb7201841cf1bfb91054eada22831e57 Merge branch 'AddingMatrixAccess' into
master
1ab8f694a40f81c0bb4747980d7c44f1f42948dc added tests to check semantic and
added tuple type to row and matrix in codegen
9fcf939fcf7181c1f9ef66d6c7a058c14f19daea Merge pull request #30 from
MichelleLu1995/file-reading
056cab3f6c398e8660059a626bdf689268f538db working files, but weird segmentation
fault on longer files
56b32d911bef9bd3f20ecf161b30a5a8553ba1a5 might work, but need to add file type
c3832e6e81d1bb9c78d3cacd5c276a5e77dfa9fe resolved merge
effa17505556d14eaab405448246a10b4840f61b added to sematic
c80147d6fde1dbf81ace3abb04f487c2542bc00c Merge pull request #29 from
MichelleLu1995/temp
34b17bc2dc9ee26277e562bb76f5d597e49bb4bb semant fix
61a67b6d35e78bd9a5062bf4cfdfaa95669556d0 removed unnecessary files
:
20d1b12d53510414a5a3d64e8f774be63a52994f deleted row ops tests
bdb5af4257079e97f2f6592fd0bf1ddb0842dd86 add matrix float adding code +
renamed/added tests
36f3f964adef5dea46d953784567602f0b4e3b53 adding matrices of intsgit add
codegen.ml semant.ml
53808caa269096070e126018662b22bd7b1bdfc4 can now subtract integer tuples of
variable length
2127045581cf176d870ec521c44ffdc4b8e9f7ba changed one of the tuple adding tests
```

```
72d10ba5851578663a3e5ffed26d9356edf4f91d can now add integer tuples of variable
lengthsgit add codegen.ml semant.ml tests/compiler_tests/test-addtuple.JSTEM
591bd7d6557d2de8ca7fbdc917e6b6c93b54d4e2 pointer successls
dd98ef65f073c285c0177e918916c6abade0f331 tuple code hard coded, but now works
9cc5c9a5d3449542305e567f85dab099cf56449e code cleanup
20a4504b2bb3aaa61f1aa4adefa442c29834b054 somehow fixed it
33e5b0759bf81439e941f4d6ebb9021fef723532 remove bad files
c648e8a51b8ed529f31f1cc7c6361eae9a6ee9c9 Matrix for loop
14795c2a1ad8e7d424144ad35e94b11366fc31ae matrix for loop
490c94d0fe7758033fd775c2b19f7a8d9441f072 can add int tuples of length 3, but
for loop not working
279573347c7b0a6b29b51fafa1368aa1f66732ee broke ints and floats but adding
tuples
4a5d5cce64603ee5f346fa3310381572664df090 having issues. don't pull
b6cc3f2475c2af05b49f74a0a141fdc8b29b07bd beginning to add vector int ops
43a48de05807274cd959459a1177d419c78b922f minor fix to tuples test
61ab00bcba27bdc70ccf275d6bc462a96c9a1ce6 added tests for adding tuples, rows,
and matrices
4b5168c71e375324e997503fdb457223cb7e6618 merge file-reading to addingmatrix
d4843e900cc8498a0d61b0b7d0221ba87c454ea3 Revert "Merge branch
'AddingMatrixAccess' into master"
25ab042fdb7201841cf1bfb91054eada22831e57 Merge branch 'AddingMatrixAccess' into
master
1ab8f694a40f81c0bb4747980d7c44f1f42948dc added tests to check semantic and
added tuple type to row and matrix in codegen
9fcf939fcf7181c1f9ef66d6c7a058c14f19daea Merge pull request #30 from
MichelleLu1995/file-reading
056cab3f6c398e8660059a626bdf689268f538db working files, but weird segmentation
fault on longer files
56b32d911bef9bd3f20ecf161b30a5a8553ba1a5 might work, but need to add file type
c3832e6e81d1bb9c78d3cacd5c276a5e77dfa9fe resolved merge
effa17505556d14eaab405448246a10b4840f61b added to sematic
c80147d6fde1dbf81ace3abb04f487c2542bc00c Merge pull request #29 from
MichelleLu1995/temp
34b17bc2dc9ee26277e562bb76f5d597e49bb4bb semant fix
61a67b6d35e78bd9a5062bf4cfdfaa95669556d0 removed unnecessary files
73e25ebe7e50e71e44781d77b25d25ecb1e5999f merge fix
a1117961d44503311561ced2a3bc2140022f8c8c just pushin some stuff lol
:
20d1b12d53510414a5a3d64e8f774be63a52994f deleted row ops tests
bdb5af4257079e97f2f6592fd0bf1ddb0842dd86 add matrix float adding code +
renamed/added tests
36f3f964adef5dea46d953784567602f0b4e3b53 adding matrices of intsgit add
codegen.ml semant.ml
53808caa269096070e126018662b22bd7b1bdfc4 can now subtract integer tuples of
variable length
2127045581cf176d870ec521c44ffdc4b8e9f7ba changed one of the tuple adding tests
72d10ba5851578663a3e5ffed26d9356edf4f91d can now add integer tuples of variable
lengthsgit add codegen.ml semant.ml tests/compiler_tests/test-addtuple.JSTEM
591bd7d6557d2de8ca7fbdc917e6b6c93b54d4e2 pointer successls
```

dd98ef65f073c285c0177e918916c6abade0f331 tuple code hard coded, but now works
9cc5c9a5d3449542305e567f85dab099cf56449e code cleanup
20a4504b2bb3aaa61f1aa4adefa442c29834b054 somehow fixed it
33e5b0759bf81439e941f4d6ebb9021fef723532 remove bad files
c648e8a51b8ed529f31f1cc7c6361eae9a6ee9c9 Matrix for loop
14795c2a1ad8e7d424144ad35e94b11366fc31ae matrix for loop
490c94d0fe7758033fd775c2b19f7a8d9441f072 can add int tuples of length 3, but for loop not working
279573347c7b0a6b29b51fafa1368aa1f66732ee broke ints and floats but adding tuples
4a5d5cce64603ee5f346fa3310381572664df090 having issues. don't pull
b6cc3f2475c2af05b49f74a0a141fdc8b29b07bd beginning to add vector int ops
43a48de05807274cd959459a1177d419c78b922f minor fix to tuples test
61ab00bcba27bdc70ccf275d6bc462a96c9a1ce6 added tests for adding tuples, rows, and matrices
4b5168c71e375324e997503fdb457223cb7e6618 merge file-reading to addingmatrix
d4843e900cc8498a0d61b0b7d0221ba87c454ea3 Revert "Merge branch 'AddingMatrixAccess' into master"
25ab042fdb7201841cf1bfb91054eada22831e57 Merge branch 'AddingMatrixAccess' into master
1ab8f694a40f81c0bb4747980d7c44f1f42948dc added tests to check semantic and added tuple type to row and matrix in codegen
9fcf939fcf7181c1f9ef66d6c7a058c14f19daea Merge pull request #30 from MichelleLu1995/file-reading
056cab3f6c398e8660059a626bdf689268f538db working files, but weird segmentation fault on longer files
56b32d911bef9bd3f20ecf161b30a5a8553ba1a5 might work, but need to add file type
c3832e6e81d1bb9c78d3cacd5c276a5e77dfa9fe resolved merge
effa17505556d14eaab405448246a10b4840f61b added to sematic
c80147d6fde1dbf81ace3abb04f487c2542bc00c Merge pull request #29 from MichelleLu1995/temp
34b17bc2dc9ee26277e562bb76f5d597e49bb4bb semant fix
61a67b6d35e78bd9a5062bf4cfdfaa95669556d0 removed unnecessary files
73e25ebe7e50e71e44781d77b25d25ecb1e5999f merge fix
a1117961d44503311561ced2a3bc2140022f8c8c just pushin some stuff lol
52c42edd98438e60fa5c79f751345e27e7532faa merge fix
b120c9610025441f3cd3b399870b9542b6791cd0 Merge pull request #28 from MichelleLu1995/semantic
27c61e368a05a43554a388463483dc3ff2408497 fix attempt
99ad3b7bef86dfdb33a78801ea515aaa8ab7a51b Revert "fixed rowlit bug"
ff5734c5758f809ea3b8057595c3f8bb89f617c5 stashing
e2287a422fc2df7c5090939d218883d2a7a8a855 the code for pointer compiles
4592345b26071cd08030884ad8d6f1d6524f0d20 matrix binops for ints/floats, add/sub
5ae40e2cacef34190b556e5ce2f66a771fb1e09c Merge branch 'pointer' into AddingMatrixAccess
729c8ce4cab27daf7842aad1c50facd94c451a1c initial stuff for pointers for matrices
4996abd522bda1b816a691167c449191b7b5dae6 fixed rowlit bug
eea1b6e292a3e03fea5161b25a4c70c6e1a31b25 Merge pull request #25 from MichelleLu1995/semantic

```
63ccf38de0ddf5cc119d790ef71686ba8b4e1a5d started adding matrix, row, and tuple
access to semant and additional tests
3ae8eab3e05a8e6e9fada917459413fe6dc2e74b string fix. comment out files
0255985d1fb487268adcc0a4ec75f3ef794e7edc testing fixed
6e280d1adefab80703467fb316cf12ec22f05f88 Merge pull request #24 from
MichelleLu1995/semantic
6f6606a47f6779794162864a2bc079898d432d62 added tuple and matrix access
07352e3a24111d9685057940875f513591a299c1 Merge pull request #23 from
MichelleLu1995/file-integration
8bd2c9a2b57d5e28e66240ae5a869eafb8e2aa31 Merge branch 'semantic' into
file-integration
753ed59de1bb714c07b1178647ad577ea83dc70f working update, not fully functional
6c13bbef1e08cf42bd2e75065f486ee169efeb4a fixed test-fun6
5801502e652c332cec21f62463b2c45d938d8880 Merge pull request #22 from
MichelleLu1995/semant
db07ef5b55f645d6212db90820a83ed4f6938d9b fixed test-ops2
a509c59ee0549018c980f9d3c1d44267820fd6c5 Changes made to semant.ml to fix fail
testings
7445672624b79bd6ec7163d1de54c5949d813780 Merge pull request #21 from
MichelleLu1995/semant
eb6e66a333698c5ae6b500deb91b260c001207bc semant checker working but w/o
matrixtyp
3bd60b542c4872b6c9886c64036e3333efb10a47 starting to add matrix access
ccf9c0017d1e2f02bb4d6013169770238bfe9e83 file integration attempt 1. errors
9383b93e3c39efdb736f9f4e0e34c5291489cebc Merge branch 'semantic' of
https://github.com/MichelleLu1995/PLT into testops2
4da0d5c70da8f03935d4cd63cdeebe75646fa5b2 Added to semant.ml
53539b8ab59fa264cf3c7c3160813c3127efa2c1 working row access
053f8794703843e46a950f7fd0301003c61d6720 deleted column type in scanner
685c1b058ee1363ef211c38a86cd162c9bd0b1a8 fixing merge conflicts
67d9985e4c5d1913838be457a8bc3147a7462e71 Merge pull request #20 from
MichelleLu1995/AddingMatrixAccess
37d438ae4d348b03d063b2ae27b553ccb16c4f8b Merge branch 'master' into
AddingMatrixAccess
f3e7eeb70438bdffdcb3fa41a035db3d86b0a61c fixing merge conflicts
b63922eee241851d3672a92991f7d4517b4df1c3 changed parser/ast to add matrix
access stuff, this branch is behind master btw
797772dc69644830f0112b3b5b33f2aea0d8ee37 added semant.ml file
dc6aadba661396e2d84f536015a5d7081a558c0e fixed commands
8b88f57b70e3540b9e488899d4077b83bcb57b72 testing script updated
14abdb1a2369d7abe09dd8fa29ce497310dc2635 Merge pull request #10 from
MichelleLu1995/compiler_tests
e8800fc220148c55277b84aa4175b8c3001500d7 Update scanner.mll
a61c17bd3fda1c7f2b3fedc0dc9cc5c14d775db4 Update codegen.ml
b946073771aa9af8e28636d80e67fe1240010fe2 fixing testing and TRUE FALSE
dd120bd12cf5dbed0febcc8f93b49aacf03bf776 merge conflicts
fb26fd93cf680912a6e57075610c7f3e887481c4 tests working except for ones with
bool
3879baea36fdceb8db0e85b7cc686f92e656fdfe changed directory of stdlib
```

```
e81c04138d31370299160f0d02504956d74582a4 starting to implement matrix access in
parser/ast
b724bc8d04db32b723d301148ced11d191151f15 Merge branch 'compiler_tests' of
https://github.com/MichelleLu1995/PLT into compiler_tests
048f01da1ea53e3e739f1c4ce9a945d663ed4904 makefile addition
19d139e4942ca7be1c47e98c0268d5a1ac7e8193 Merge branch 'compiler_tests' of
https://github.com/MichelleLu1995/PLT into compiler_tests
a9f0e711512680af3a2729a4fdb32db7a2d7691a add1 test working
7262d6b64c2741458f6affc3f05dea77f4425834 Merge branch 'compiler_tests' of
https://github.com/MichelleLu1995/PLT into compiler_tests
b009d67f5315f23e6b6631f0b35a664bd043ef09 testing fixes
e39a08d5bf70345768a12701e93c6c6a544630cb cleans .err files
6161d5e09a91c311e07d0a8f17f2b72dcac63037 added -c to make testall run
3d759254322b16bf90cbbc9f2c443fa3ea0a55ee Merge pull request #9 from
MichelleLu1995/newline
5ca5606d2c40d20b14baa327a0044449d362dd15 prints newline string
1ae869ef1a52acf183f4a1b5cec8ce206c811d55 Merge branch 'compiler_tests' of
https://github.com/MichelleLu1995/PLT into compiler_tests
41b97c412f11a3ff9b115a3ef19c57c1633a1333 added stdlib to testall not working
495030a972b8ff789315eb85b6de01653b59c4af Merge branch 'compiler_tests' of
https://github.com/MichelleLu1995/PLT into compiler_tests
bd9f447bc4a9fa7f6758bed9b93e8b992f471241 scanner parser testing working
686b3660fede1140a4fe6deffbb2111aa9354a9d Merge branch 'compiler_tests' of
https://github.com/MichelleLu1995/PLT into compiler_tests
7889e9f01aaaa6669afa19bf8e41af52be729fc2 edited t/f to T/F
057ece17a7c697819602bf534862e9e7ac598d9d Merge branch 'compiler_tests' of
https://github.com/MichelleLu1995/PLT into compiler_tests
2dbb416626a134e78c077b67779619653fe5619b script attempt
f5e40fc6f25cd4844b1db645b8c36457f2b7178e added more tests
af3f68cf3d2b0e71e4f7539e5085e937405f6b52 Merge branch 'compiler_tests' of
https://github.com/MichelleLu1995/PLT into compiler_tests
13e664225043fb40a40b4c793d330c4a90b7fce0 added more tests
ea8a9f71af6af52461003a884e33f9b3cd020ca7 Merge branch 'compiler_tests' of
https://github.com/MichelleLu1995/PLT into compiler_tests
300f966fb389b235611be74c75108d08e737849d compiler fail tests with the .JSTEM
extension
feb1b2e2330fccd586078fcceb7f9e707707767a Merge branch 'master' of
https://github.com/MichelleLu1995/PLT into compiler_tests
6b4cef099e61d9c4e16deba06a3830054ee2bf4e Merge branch 'compiler_tests' of
https://github.com/MichelleLu1995/PLT into compiler_tests
9e0fdde2f3343c131eb91ba8ed73ee38b9909567 Merge branch 'compiler_tests' of
https://github.com/MichelleLu1995/PLT into compiler_tests
c072b4dd283840e242797160dd578b72d009cdf3 Delete fail-while2.mc
596b50b98a5dfca3830ebba8b334a4b2a6fcf5db Delete fail-while1.mc
528916eec1bc645ab7025e6a3d76fa68626eb637 Delete fail-return2.mc
5142bed9346389356cda684db705ec6190165c35 Delete fail-return1.mc
5142bed9346389356cda684db705ec6190165c35 Delete fail-return1.mc
fb6447892aa25222268612d614cc5d8fb692b963 Delete fail-if3.mc
2934d90273ba5eff3b3315fb7ebbea2c27ad53e7 Merge branch 'compiler_tests' of
https://github.com/MichelleLu1995/PLT into compiler_tests
```

892aaab29b4771c2178fd8f232b28f91df949212 Delete fail-if2.mc
e2721e8f90ee5df8d562760f85b7d30909a7f4fb Delete fail-nomain.mc
c3fbc4b6bf59ec2f88fcccab67d124d955bfca9c Delete fail-if1.mc
59af9f3227117e2e56b7429ce60c3411c0a8d271 Delete fail-global2.mc
bf9047d85c506b73f26966b0255ff805cf22774e Delete fail-global1.mc
a586458551ff3e21873c17c524779e3ec3b55479 Delete fail-func9.mc
e6e93880a6c7056e5b74581d2b423443558b6add Delete fail-func8.mc
f17057b202dea688fd1cd81685353468b77e347e Delete fail-func7.mc
3eca03f2560118508880dbdf6f147d946cdf5ef2 Delete fail-func6.mc
e5e0ea073a31fdf01fbf2e9ae3a11c3a534a7615 Delete fail-func5.mc
09804a5f03963c066aa5b68366a660a8bc5dc7ce Merge branch 'compiler_tests' of
https://github.com/MichelleLu1995/PLT into compiler_tests
5c894e9fbba4daef24e27ac0e2e2c2ed4bc6525d Delete fail-func4.mc
dd3c034fdcb220e09bdb47a2b8651c150d39be5f Delete fail-func3.mc
b98af7cba1c804b316b694de5d9882a9807d93d8 Delete fail-func2.mc
11cb79503005c23ad60d5fe7db6b0acb392fd73e added some tests
efbc4228d4430dce7ed46cb330039141878912ea Delete fail-func1.mc
c0284e70c9813ede5bbe0a8895b9a91bcfdb3657 Delete fail-for5.mc
648e6798eec1f94ecd266209a07ba3e5fd51859b Delete fail-for4.mc
4a7646baf5fbb0b53883c88e4df0f1acb755fd87 Delete fail-for3.mc
e4168605d95fb9ee8b720bed7749ea9b2b8e03b9 Delete fail-for2.mc
6238e5789ffc845db035aa32716e9f65bf9081e8 Delete fail-for1.mc
f22c1a28b0b44dfba35b7da15dd58e4fc4bc3a6a Delete fail-expr2.mc
7916def871e04babc44acc7f180fa56d6943a822 Delete fail-expr1.mc
c58a8534832c11e354c80ae5bcac412256eb1d9a Delete fail-dead2.mc
f4cd030908b64ec9edf87a686b502998896247c7 Delete fail-dead1.mc
a05f6c8c7a9cd8505f7ef0fec37b5baef0ee8fa7 Delete fail-assign3.mc
7ace5243a476285728fdedc32144a0009b0dc33e Delete fail-assign2.mc
ff44420b3e8ce07b525b0a4f2bb010c1b951eea7 Delete fail-assign1.mc
c1c6393fdda57709fb8bad8b440b596f5847cd72 Merge branch 'compiler_tests' of
https://github.com/MichelleLu1995/PLT into compiler_tests
678627f52b9fe5dd628938c633f593a37300e0a1 fail tests for compiler
c6799acb3bdca03de0662b56c918705a55e50400 included fail_assign1 and fail_assign2
4dc986c52e6727b3ebc77f456cb3e2507f460496 created a new directory where we will
take the microc test suite and rewrite the tests to be in the JSTEM language
9beb8d1519e6ad63362e4bb2193eaec26e19196b changed string_literal regexp
5c7b7964e5e5257fba9caa22f4a32c4baf285973 Merge branch 'master' of
https://github.com/MichelleLu1995/PLT
32cddb190845f5a6fcc694ab0571017713c4681b git pulling
b00cbfdd7e053a1e141fe4c5281a9401a15e6c96 working hello world
f332b1831ac0a8126f357a95f8db0ea1bd9d8f46 can compile to LLVM IR
d364e1f837ca482c55d1c9c4571335d34a78f729 Merge pull request #8 from
MichelleLu1995/codegen-makefile
1096c6d94477e77a27514bea82e5096617186dc5 project building
08987157e9bb261b5b49c02b56c14ff555a32460 Merge pull request #7 from
MichelleLu1995/codegen-makefile
c193a8624ef233505b10a596fc47630adb012a61 build codegen
4118ad25cf826972b452375a22a054d9585fe8e1 Makefile for codegen compiling
39fd285fa7aa60331445290693b02a733a5f2c29 Merge pull request #6 from
MichelleLu1995/codegen-syntax

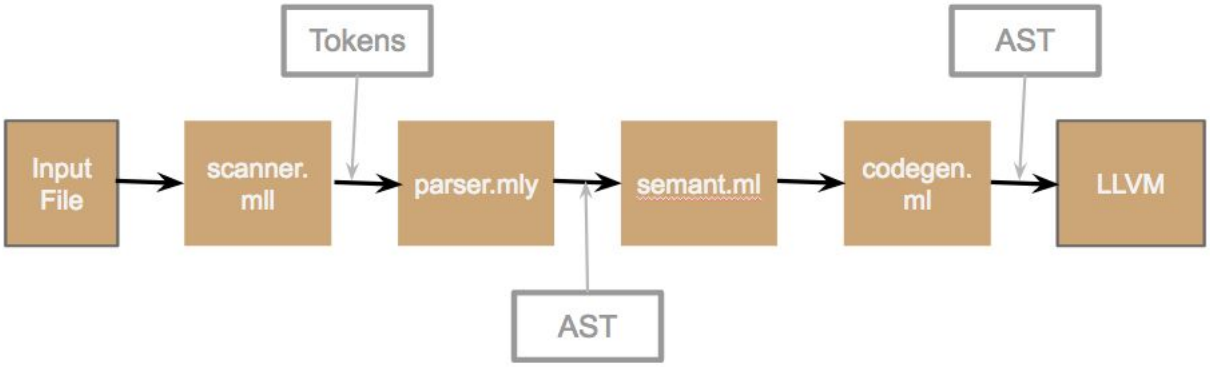
```
6847666989dc886951d346800171ba02abaad719 fixed minor syntax issues
b84202c4daf323bbafd229d14e6b37dda2d7df3a Merge pull request #5 from
MichelleLu1995/codegen-helloworld
5cbb3f3b815def0b0571724cefea5760b67fd40b Merge branch 'master' into
codegen-helloworld
044e973f1489f77157c46a9ec7f4a970b98e8959 codegen necessary for hello world
5fbbfd66280067953684c699057b928a757dc982 commented out non-hello world codegen
ceec24e163e8ee279d55231d274730c07c8fa54d Merge branch 'master' of
https://github.com/MichelleLu1995/PLT
668579751368ca594d83ec36552416312af48360 parse 'Hello World' test
9c28aa5d0adc414efca05cf1772e1e07f5d32388 Merge branch 'master' of
https://github.com/MichelleLu1995/PLT
0b8749406ceac7367b3fac5b3f7727f79bc2b8c3 parser tests working. travis should
work
dec8914bcd6ab1d68435e18d06b2798c9a7e1837 Merge pull request #4 from
MichelleLu1995/emily
6c0e516e4bbf8c840a07459f69fbd9b3dc959643 fixed merge conflicts
ccf6f0610296968047cc2effce1d4902589f4a50 first codegen commit
7387735376b59e6a0285faec0838e98daa9bac47 removed columns
a287dc6817f8fe55d1593edcdbd478fd4073937a test suit cleanup and execution
f0390194d288a2f36c90c3557190b8af4063a507 resolved merge conflicts
8496b77d4438cc25a231646b990042a11d7291de Merge branch 'ASTfirstmodifications'
f365c6f321bda1a6449d237c7cbb2eca4a9ccc41 no shift/reduce reduce/reduce
conflicts
31f3b1c91202cfc699d756a6cc9ae4d5286cab49 fixed tuple reduce reduce conflict
b08d24a0f94957d336e87793a4e2ac9c699a23f4 Merge branch 'ASTfirstmodifications'
20eefbd1a229e446cf1a111ef5790e614d55f130 changed matrix structure, commented
out all column
8367a75166567f1756a1124cb9a82ec5caf10693 Merge pull request #3 from
MichelleLu1995/ASTfirstmodifications
32e00fc359df406aaf3a677e76df4d0b4ca2d04a changed string_of_matrix to match ours
703024b1cb904fa828c34ea677a44b9efd7c071f Merge pull request #2 from
MichelleLu1995/ASTfirstmodifications
f7732619089eacc8b2c0f257d6249f39831a479e debugged ast and parser
c25c23a0746dfa7a56a73f619d5c1104cea459a7 travis attempt
5bb000ffcf78c8b6ab58ddd603d81eb6b2e363dd working on AST and parser
2b788e5c372266a94bdc80156144d1ebca748763 Merge branch 'master' of
https://github.com/MichelleLu1995/PLT
4bcc6a88d90457ea3b86fdac87c7d2c1144bb0e0 changes to parser while working on AST
b8e7d87069efd87db0582be8bd82da5ca2f2b89e more tests for the scanner
ccaffa445fbc37833d4d7bd8ebb9b7c02dac8f9a office hours stuff -- in progress
354577e79eee1fcc2d4845d4f6d1ec9250fb2ae1 travis fixes
56ece793aac962db370ba812e613e760f1472b09 merge testing folders
13ded292702f983670e817a9af7a59994b26257e Merge pull request #1 from
MichelleLu1995/testing
8432f394539afaa6fa3f75cca02134eab3e13d94 general testing setup, travis and
makefiles
aec448dc8f0aba0369a9d27633a8a05605ffa833 edited ast and added test files
4d562490b7ecc3d024248dea9d9639e5eb17a3d3 tests for conditionals, datatypes, and
function keywords
```

```
9f92c6545a8c72dda28cbe31c542fec4e6cd641f tokenizer
2e706baccc7ab1dce4fce082f6274edb85ac787c remove microc
c98154fdbcd478a8f3f754434489bcc75ef2e9ee added BAR to scanner and updated
parser
0304389fe51bf34f5c1dff374131f25659cd6a9a scanner parses correct tokens for
Hello World
1287eb8d702dd34f92ac6d819d4daafd4a793e73 travis file initial
10a5449be2fc488051a84b8476d6257d851e261c Added new tokens to scanner.mll
2c8eee685d13506e1c8baec8a367d76381ace40b microc files
c96bbd9e30ee3a1215cd55c0a3a7574ddc7dd057 first commit
```

5. Architectural Design

a. Architecture Diagram

This diagram depicts an overarching view of the architecture of the programming language:



The architecture follows a basic flow with the input file being a .JSTEM file. It also compiles to LLVM. An important note is that we do not have an sast. Type checking all happens happens in semant.ml.

b. Scanner

This module takes in a .JSTEM file and generates tokens and ignores the whitespace and comments in the JSTEM language. These tokens include keywords, operators, literals, etc. Once the tokens are created, they are then passed to the parser.

c. Parser

This module produces an Abstract Syntax Tree (AST) from the tokens made from the scanner. The parser also indicates how various types (matrix_pointer_typ, typle_typ, etc.) that are used should be read in. If the sequence of tokens are not about to be parsed, an error is thrown.

d. AST

This module represents the program after the parser. The JSTEM AST will also return some errors to inform the user what kind of errors they may be facing if some of the code they are writing violates the syntax code.

e. Semantic Analysis

This module ensures that a source program or file adheres to rules such as syntax for the JSTEM language. The semantic analysis does this check by looking at the AST. Similarly to the AST, the semant.ml also tells the programmer if more logically the code they are writing doesn't make sense. It will return more logically-based errors to help the user debug. It will tell the user what kind of object it expects to return and inform them what they are currently receiving.

f. LLVM Code Generation

The codegen module builds the LLVM instructions into a file. Codegen.ml then uses the AST passed into it by analyzer.

6. Test Plan

As previously stated, we modeled our tests off of the microc tests. We included a `./compiler_testing.sh` script to run through all the tests that we had. This script allowed for automation and for all tests to be checked at once. These tests were hosted in a "tests" directory. Every time a new feature was included, we wrote new tests to see if the features were working and ran the whole test suite to see if any of the features that were implemented previously had been broken while implementing the new features. This helped with fixing bugs and identifying areas where our code conflicted.

With unit testing, we also would run the "menhir" command on the parser to identify whether or not there were shift reduce conflicts and find out whether or not the parser would accept the given input tokens.

With integration testing, we also wrote a pretty printer to ensure that the scanner, parser, and AST were working. This was also modeled off of microc's pretty printer. We then used a "diff" command to compare and see if the output that we got was expected.

We also implemented continuous integration using Travis CI. Every time a commit was pushed, it automatically runs all the tests and checks if the tests still pass. This was helpful as we made

commits and pushed to the master because we would know if any new code broke tests that were previously working.

a. Test Example 1

This first example shows how matrices and rows can be declared with the primitives of int, float, and tuple. This example also shows are specific print functions are employed to print certain data structures and make use of the .length and .width functions in order to know when to stop printing or know how much should be printed out. This example was chosen because it shows a robust representation of what can be done with tuples and matrices.

a.1 Example 1 in Native Language

test-tuplematrixrowprinting.JSTEM

```
def int main() {

        int[4] int_row;
        int[2][4] int_matrix;

        float[3] float_row;
        float[2][3] float_matrix;

        int(%3%) t;
        int(%3%)[2] tuple_row;
        int(%3%)[2][2] tuple_matrix;

        int_row = [1,2,3,4];
        int_matrix = {% 1,2,3,4 | 5,6,7,8 %};

        float_row = [1.1,2.2,3.3];
        float_matrix = {% 1.1,2.2,3.3 | 4.4,5.5,6.6 %};

        t = (%1,2,3%);
        tuple_row = [(%1,2,3%),(%3,4,5%)];
        tuple_matrix = {% (%1,2,3%), (%4,5,6%) | (%7,8,9%), (%10,11,12%) %};

        prints("ints: \n");
        print_rowi($int_row, int_row.length);
        prints("\n");
        print_matrixi($$int_matrix, int_matrix.length, int_matrix.width);
        prints("\n");

        prints("floats: \n");
        print_rowf($float_row, float_row.length);
        prints("\n");
        print_matrixf($$float_matrix, float_matrix.length, float_matrix.width);
```

```
        prints("\n");

        prints("tuples: \n");
        print_tuple(t);
        prints("\n");
        print_rowt($tuple_row, tuple_row.length);
        prints("\n");
        print_matrixt($$tuple_matrix, tuple_matrix.length, tuple_matrix.width);
        prints("\n");

        return 0;

}
```

test-tuplematrixrowprinting.out

```
ints:
[ 1, 2, 3, 4 ]
{ 1, 2, 3, 4
  5, 6, 7, 8 }
floats
[ 1.100000, 2.200000, 3.300000 ]
{ 1.100000, 2.200000, 3.300000
  4.400000, 5.500000, 6.600000 }
tuples:
( 1, 2, 3 )
[ ( 1, 2, 3 ), ( 3, 4, 5 ) ]
{ ( 1, 2, 3 ), ( 4, 5, 6 )
  ( 7, 8, 9 ), ( 10, 11, 12 ) }
```

a.2 Example 1 in Target Language

```
; ModuleID = 'JSTEM'

@fmt = private unnamed_addr constant [3 x i8] c"%d\00"
@fmt1 = private unnamed_addr constant [3 x i8] c"%f\00"
@fmt2 = private unnamed_addr constant [3 x i8] c"%c\00"
@fmt3 = private unnamed_addr constant [3 x i8] c"%d\00"
@fmt4 = private unnamed_addr constant [3 x i8] c"%f\00"
@fmt5 = private unnamed_addr constant [3 x i8] c"%c\00"
@fmt6 = private unnamed_addr constant [3 x i8] c"%d\00"
@fmt7 = private unnamed_addr constant [3 x i8] c"%f\00"
@fmt8 = private unnamed_addr constant [3 x i8] c"%c\00"
@.str = private unnamed_addr constant [3 x i8] c"{ \00"
@.str9 = private unnamed_addr constant [3 x i8] c", \00"
@.str10 = private unnamed_addr constant [4 x i8] c"\0A  \00"
```

```
@.str11 = private unnamed_addr constant [3 x i8] c" }\00"
@fmt12 = private unnamed_addr constant [3 x i8] c"%d\00"
@fmt13 = private unnamed_addr constant [3 x i8] c"%f\00"
@fmt14 = private unnamed_addr constant [3 x i8] c"%c\00"
@.str15 = private unnamed_addr constant [3 x i8] c"{ \00"
@.str16 = private unnamed_addr constant [3 x i8] c", \00"
@.str17 = private unnamed_addr constant [4 x i8] c"\0A  \00"
@.str18 = private unnamed_addr constant [3 x i8] c" }\00"
@fmt19 = private unnamed_addr constant [3 x i8] c"%d\00"
@fmt20 = private unnamed_addr constant [3 x i8] c"%f\00"
@fmt21 = private unnamed_addr constant [3 x i8] c"%c\00"
@.str22 = private unnamed_addr constant [3 x i8] c"{ \00"
@.str23 = private unnamed_addr constant [3 x i8] c", \00"
@.str24 = private unnamed_addr constant [4 x i8] c"\0A  \00"
@.str25 = private unnamed_addr constant [3 x i8] c" }\00"
@fmt26 = private unnamed_addr constant [3 x i8] c"%d\00"
@fmt27 = private unnamed_addr constant [3 x i8] c"%f\00"
@fmt28 = private unnamed_addr constant [3 x i8] c"%c\00"
@.str29 = private unnamed_addr constant [3 x i8] c"[ \00"
@.str30 = private unnamed_addr constant [3 x i8] c", \00"
@.str31 = private unnamed_addr constant [3 x i8] c" ]\00"
@fmt32 = private unnamed_addr constant [3 x i8] c"%d\00"
@fmt33 = private unnamed_addr constant [3 x i8] c"%f\00"
@fmt34 = private unnamed_addr constant [3 x i8] c"%c\00"
@.str35 = private unnamed_addr constant [3 x i8] c"[ \00"
@.str36 = private unnamed_addr constant [3 x i8] c", \00"
@.str37 = private unnamed_addr constant [3 x i8] c" ]\00"
@fmt38 = private unnamed_addr constant [3 x i8] c"%d\00"
@fmt39 = private unnamed_addr constant [3 x i8] c"%f\00"
@fmt40 = private unnamed_addr constant [3 x i8] c"%c\00"
@.str41 = private unnamed_addr constant [3 x i8] c"[ \00"
@.str42 = private unnamed_addr constant [3 x i8] c", \00"
@.str43 = private unnamed_addr constant [3 x i8] c" ]\00"
@fmt44 = private unnamed_addr constant [3 x i8] c"%d\00"
@fmt45 = private unnamed_addr constant [3 x i8] c"%f\00"
@fmt46 = private unnamed_addr constant [3 x i8] c"%c\00"
@.str47 = private unnamed_addr constant [3 x i8] c"( \00"
@.str48 = private unnamed_addr constant [3 x i8] c", \00"
@.str49 = private unnamed_addr constant [3 x i8] c" )\00"
@fmt50 = private unnamed_addr constant [3 x i8] c"%d\00"
@fmt51 = private unnamed_addr constant [3 x i8] c"%f\00"
@fmt52 = private unnamed_addr constant [3 x i8] c"%c\00"
@.str53 = private unnamed_addr constant [8 x i8] c"ints: \0A\00"
@.str54 = private unnamed_addr constant [2 x i8] c"\0A\00"
@.str55 = private unnamed_addr constant [2 x i8] c"\0A\00"
@.str56 = private unnamed_addr constant [10 x i8] c"floats: \0A\00"
@.str57 = private unnamed_addr constant [2 x i8] c"\0A\00"
@.str58 = private unnamed_addr constant [2 x i8] c"\0A\00"
@.str59 = private unnamed_addr constant [10 x i8] c"tuples: \0A\00"
```

```
@.str60 = private unnamed_addr constant [2 x i8] c"\0A\00"
@.str61 = private unnamed_addr constant [2 x i8] c"\0A\00"
@.str62 = private unnamed_addr constant [2 x i8] c"\0A\00"

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

declare i8* @puts(i8*, ...)

declare i8* @fopen(i8*, i8*)

declare i32 @fclose(i8*)

declare i32 @fread(i8*, i32, i32, i8*)

declare i8* @fgets(i8*, i32, i8*)

declare i32 @fwrite(i8*, i32, i32, i8*)

declare i32 @strlen(i8*)

declare i32 @atoi(i8*)

declare i8* @sprintf(i8*, i8*, i32)

declare i8* @calloc(i32, i32)

declare i8* @strtok(i8*, i8*)

declare i8* @strstr(i8*, i8*)

declare i32 @strcmp(i8*, i8*)

define double* @add_rowf(double* %oldm, double* %newm_ptr, double* %row, i32 %len, i32 %wid)
{
entry:
  %oldm1 = alloca double*
  store double* %oldm, double** %oldm1
  %newm_ptr2 = alloca double*
  store double* %newm_ptr, double** %newm_ptr2
  %row3 = alloca double*
  store double* %row, double** %row3
  %len4 = alloca i32
  store i32 %len, i32* %len4
  %wid5 = alloca i32
  store i32 %wid, i32* %wid5
  %i = alloca i32
  %j = alloca i32
  %len_plus_1 = alloca i32
  %offset1 = alloca i32
```

```
  %offset = alloca i32
  %newm = alloca double*
  %_i = alloca i32
  %len6 = load i32* %len4
  %wid7 = load i32* %wid5
  %tmp = mul i32 %len6, %wid7
  store i32 %tmp, i32* %offset1
  %offset18 = load i32* %offset1
  %wid9 = load i32* %wid5
  %tmp10 = add i32 %offset18, %wid9
  store i32 %tmp10, i32* %offset
  %len11 = load i32* %len4
  %tmp12 = add i32 %len11, 1
  store i32 %tmp12, i32* %len_plus_1
  %newm_ptr13 = load double** %newm_ptr2
  store double* %newm_ptr13, double** %newm
  store i32 0, i32* %i
  br label %while

while:                                            ; preds = %merge, %entry
  %i37 = load i32* %i
  %offset38 = load i32* %offset
  %tmp39 = icmp slt i32 %i37, %offset38
  br i1 %tmp39, label %while_body, label %merge40

while_body:                                       ; preds = %while
  %i14 = load i32* %i
  %offset115 = load i32* %offset1
  %tmp16 = icmp slt i32 %i14, %offset115
  br i1 %tmp16, label %then, label %else

merge:                                            ; preds = %else, %then
  %i35 = load i32* %i
  %tmp36 = add i32 %i35, 1
  store i32 %tmp36, i32* %i
  br label %while

then:                                             ; preds = %while_body
  %newm17 = load double** %newm
  %oldm18 = load double** %oldm1
  %oldm19 = load double* %oldm18
  store double %oldm19, double* %newm17
  %oldm20 = getelementptr inbounds double** %oldm1, i32 0
  %oldm21 = load double** %oldm20
  %oldm22 = getelementptr inbounds double* %oldm21, i32 1
  store double* %oldm22, double** %oldm1
  %newm23 = getelementptr inbounds double** %newm, i32 0
  %newm24 = load double** %newm23
  %newm25 = getelementptr inbounds double* %newm24, i32 1
```

```
  store double* %newm25, double** %newm
  br label %merge

else:                                             ; preds = %while_body
  %newm26 = load double** %newm
  %row27 = load double** %row3
  %row28 = load double* %row27
  store double %row28, double* %newm26
  %row29 = getelementptr inbounds double** %row3, i32 0
  %row30 = load double** %row29
  %row31 = getelementptr inbounds double* %row30, i32 1
  store double* %row31, double** %row3
  %newm32 = getelementptr inbounds double** %newm, i32 0
  %newm33 = load double** %newm32
  %newm34 = getelementptr inbounds double* %newm33, i32 1
  store double* %newm34, double** %newm
  br label %merge

merge40:                                          ; preds = %while
  %newm_ptr41 = load double** %newm_ptr2
  ret double* %newm_ptr41
}

define i32* @add_rowi(i32* %oldm, i32* %newm_ptr, i32* %row, i32 %len, i32 %wid) {
entry:
  %oldm1 = alloca i32*
  store i32* %oldm, i32** %oldm1
  %newm_ptr2 = alloca i32*
  store i32* %newm_ptr, i32** %newm_ptr2
  %row3 = alloca i32*
  store i32* %row, i32** %row3
  %len4 = alloca i32
  store i32 %len, i32* %len4
  %wid5 = alloca i32
  store i32 %wid, i32* %wid5
  %i = alloca i32
  %j = alloca i32
  %len_plus_1 = alloca i32
  %offset1 = alloca i32
  %offset = alloca i32
  %newm = alloca i32*
  %_i = alloca i32
  %len6 = load i32* %len4
  %wid7 = load i32* %wid5
  %tmp = mul i32 %len6, %wid7
  store i32 %tmp, i32* %offset1
  %offset18 = load i32* %offset1
  %wid9 = load i32* %wid5
  %tmp10 = add i32 %offset18, %wid9
```

```
  store i32 %tmp10, i32* %offset
  %len11 = load i32* %len4
  %tmp12 = add i32 %len11, 1
  store i32 %tmp12, i32* %len_plus_1
  %newm_ptr13 = load i32** %newm_ptr2
  store i32* %newm_ptr13, i32** %newm
  store i32 0, i32* %i
  br label %while

while:                                            ; preds = %merge, %entry
  %i37 = load i32* %i
  %offset38 = load i32* %offset
  %tmp39 = icmp slt i32 %i37, %offset38
  br i1 %tmp39, label %while_body, label %merge40

while_body:                                       ; preds = %while
  %i14 = load i32* %i
  %offset115 = load i32* %offset1
  %tmp16 = icmp slt i32 %i14, %offset115
  br i1 %tmp16, label %then, label %else

merge:                                            ; preds = %else, %then
  %i35 = load i32* %i
  %tmp36 = add i32 %i35, 1
  store i32 %tmp36, i32* %i
  br label %while

then:                                             ; preds = %while_body
  %newm17 = load i32** %newm
  %oldm18 = load i32** %oldm1
  %oldm19 = load i32* %oldm18
  store i32 %oldm19, i32* %newm17
  %oldm20 = getelementptr inbounds i32** %oldm1, i32 0
  %oldm21 = load i32** %oldm20
  %oldm22 = getelementptr inbounds i32* %oldm21, i32 1
  store i32* %oldm22, i32** %oldm1
  %newm23 = getelementptr inbounds i32** %newm, i32 0
  %newm24 = load i32** %newm23
  %newm25 = getelementptr inbounds i32* %newm24, i32 1
  store i32* %newm25, i32** %newm
  br label %merge

else:                                             ; preds = %while_body
  %newm26 = load i32** %newm
  %row27 = load i32** %row3
  %row28 = load i32* %row27
  store i32 %row28, i32* %newm26
  %row29 = getelementptr inbounds i32** %row3, i32 0
  %row30 = load i32** %row29
```

```
  %row31 = getelementptr inbounds i32* %row30, i32 1
  store i32* %row31, i32** %row3
  %newm32 = getelementptr inbounds i32** %newm, i32 0
  %newm33 = load i32** %newm32
  %newm34 = getelementptr inbounds i32* %newm33, i32 1
  store i32* %newm34, i32** %newm
  br label %merge

merge40:                                          ; preds = %while
  %newm_ptr41 = load i32** %newm_ptr2
  ret i32* %newm_ptr41
}

define void @print_matrixt([3 x i32]* %m, i32 %len, i32 %wid) {
entry:
  %m1 = alloca [3 x i32]*
  store [3 x i32]* %m, [3 x i32]** %m1
  %len2 = alloca i32
  store i32 %len, i32* %len2
  %wid3 = alloca i32
  store i32 %wid, i32* %wid3
  %i = alloca i32
  %j = alloca i32
  %len_minus_1 = alloca i32
  %wid_minus_1 = alloca i32
  %_i = alloca i32
  %len4 = load i32* %len2
  %tmp = sub i32 %len4, 1
  store i32 %tmp, i32* %len_minus_1
  %wid5 = load i32* %wid3
  %tmp6 = sub i32 %wid5, 1
  store i32 %tmp6, i32* %wid_minus_1
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str, i32 0,
i32 0))
  store i32 0, i32* %i
  br label %while

while:                                            ; preds = %merge30, %entry
  %i33 = load i32* %i
  %len34 = load i32* %len2
  %tmp35 = icmp slt i32 %i33, %len34
  br i1 %tmp35, label %while_body, label %merge36

while_body:                                       ; preds = %while
  store i32 0, i32* %j
  br label %while7

while7:                                           ; preds = %merge, %while_body
  %j27 = load i32* %j
```

```
  %wid28 = load i32* %wid3
  %tmp29 = icmp slt i32 %j27, %wid28
  br i1 %tmp29, label %while_body8, label %merge30

while_body8:                                      ; preds = %while7
  %m9 = load [3 x i32]** %m1
  %m10 = load [3 x i32]* %m9
  call void @print_tuple([3 x i32] %m10)
  %j11 = load i32* %j
  %wid_minus_112 = load i32* %wid_minus_1
  %tmp13 = icmp slt i32 %j11, %wid_minus_112
  br i1 %tmp13, label %then, label %else

merge:                                            ; preds = %merge18, %then
  %m22 = getelementptr inbounds [3 x i32]** %m1, i32 0
  %m23 = load [3 x i32]** %m22
  %m24 = getelementptr inbounds [3 x i32]* %m23, i32 1
  store [3 x i32]* %m24, [3 x i32]** %m1
  %j25 = load i32* %j
  %tmp26 = add i32 %j25, 1
  store i32 %tmp26, i32* %j
  br label %while7

then:                                             ; preds = %while_body8
  %printf14 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str9, i32
0, i32 0))
  br label %merge

else:                                             ; preds = %while_body8
  %i15 = load i32* %i
  %len_minus_116 = load i32* %len_minus_1
  %tmp17 = icmp slt i32 %i15, %len_minus_116
  br i1 %tmp17, label %then19, label %else21

merge18:                                          ; preds = %else21, %then19
  br label %merge

then19:                                           ; preds = %else
  %printf20 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @.str10,
i32 0, i32 0))
  br label %merge18

else21:                                           ; preds = %else
  br label %merge18

merge30:                                          ; preds = %while7
  %i31 = load i32* %i
  %tmp32 = add i32 %i31, 1
  store i32 %tmp32, i32* %i
```

```
  br label %while

merge36:                                          ; preds = %while
  %printf37 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str11,
i32 0, i32 0))
  ret void
}

define void @print_matrixf(double* %m, i32 %len, i32 %wid) {
entry:
  %m1 = alloca double*
  store double* %m, double** %m1
  %len2 = alloca i32
  store i32 %len, i32* %len2
  %wid3 = alloca i32
  store i32 %wid, i32* %wid3
  %i = alloca i32
  %j = alloca i32
  %len_minus_1 = alloca i32
  %wid_minus_1 = alloca i32
  %_i = alloca i32
  %len4 = load i32* %len2
  %tmp = sub i32 %len4, 1
  store i32 %tmp, i32* %len_minus_1
  %wid5 = load i32* %wid3
  %tmp6 = sub i32 %wid5, 1
  store i32 %tmp6, i32* %wid_minus_1
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str15, i32
0, i32 0))
  store i32 0, i32* %i
  br label %while

while:                                            ; preds = %merge31, %entry
  %i34 = load i32* %i
  %len35 = load i32* %len2
  %tmp36 = icmp slt i32 %i34, %len35
  br i1 %tmp36, label %while_body, label %merge37

while_body:                                       ; preds = %while
  store i32 0, i32* %j
  br label %while7

while7:                                           ; preds = %merge, %while_body
  %j28 = load i32* %j
  %wid29 = load i32* %wid3
  %tmp30 = icmp slt i32 %j28, %wid29
  br i1 %tmp30, label %while_body8, label %merge31

while_body8:                                      ; preds = %while7
```

```
  %m9 = load double** %m1
  %m10 = load double* %m9
  %printf11 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @fmt13, i32
0, i32 0), double %m10)
  %j12 = load i32* %j
  %wid_minus_113 = load i32* %wid_minus_1
  %tmp14 = icmp slt i32 %j12, %wid_minus_113
  br i1 %tmp14, label %then, label %else

merge:                                            ; preds = %merge19, %then
  %m23 = getelementptr inbounds double** %m1, i32 0
  %m24 = load double** %m23
  %m25 = getelementptr inbounds double* %m24, i32 1
  store double* %m25, double** %m1
  %j26 = load i32* %j
  %tmp27 = add i32 %j26, 1
  store i32 %tmp27, i32* %j
  br label %while7

then:                                             ; preds = %while_body8
  %printf15 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str16,
i32 0, i32 0))
  br label %merge

else:                                             ; preds = %while_body8
  %i16 = load i32* %i
  %len_minus_117 = load i32* %len_minus_1
  %tmp18 = icmp slt i32 %i16, %len_minus_117
  br i1 %tmp18, label %then20, label %else22

merge19:                                          ; preds = %else22, %then20
  br label %merge

then20:                                           ; preds = %else
  %printf21 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @.str17,
i32 0, i32 0))
  br label %merge19

else22:                                           ; preds = %else
  br label %merge19

merge31:                                          ; preds = %while7
  %i32 = load i32* %i
  %tmp33 = add i32 %i32, 1
  store i32 %tmp33, i32* %i
  br label %while

merge37:                                          ; preds = %while
```

```
  %printf38 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str18,
i32 0, i32 0))
  ret void
}

define void @print_matrixi(i32* %m, i32 %len, i32 %wid) {
entry:
  %m1 = alloca i32*
  store i32* %m, i32** %m1
  %len2 = alloca i32
  store i32 %len, i32* %len2
  %wid3 = alloca i32
  store i32 %wid, i32* %wid3
  %i = alloca i32
  %j = alloca i32
  %len_minus_1 = alloca i32
  %wid_minus_1 = alloca i32
  %_i = alloca i32
  %len4 = load i32* %len2
  %tmp = sub i32 %len4, 1
  store i32 %tmp, i32* %len_minus_1
  %wid5 = load i32* %wid3
  %tmp6 = sub i32 %wid5, 1
  store i32 %tmp6, i32* %wid_minus_1
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str22, i32
0, i32 0))
  store i32 0, i32* %i
  br label %while

while:                                            ; preds = %merge31, %entry
  %i34 = load i32* %i
  %len35 = load i32* %len2
  %tmp36 = icmp slt i32 %i34, %len35
  br i1 %tmp36, label %while_body, label %merge37

while_body:                                       ; preds = %while
  store i32 0, i32* %j
  br label %while7

while7:                                           ; preds = %merge, %while_body
  %j28 = load i32* %j
  %wid29 = load i32* %wid3
  %tmp30 = icmp slt i32 %j28, %wid29
  br i1 %tmp30, label %while_body8, label %merge31

while_body8:                                      ; preds = %while7
  %m9 = load i32** %m1
  %m10 = load i32* %m9
```

```
  %printf11 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @fmt19, i32
0, i32 0), i32 %m10)
  %j12 = load i32* %j
  %wid_minus_113 = load i32* %wid_minus_1
  %tmp14 = icmp slt i32 %j12, %wid_minus_113
  br i1 %tmp14, label %then, label %else

merge:                                            ; preds = %merge19, %then
  %m23 = getelementptr inbounds i32** %m1, i32 0
  %m24 = load i32** %m23
  %m25 = getelementptr inbounds i32* %m24, i32 1
  store i32* %m25, i32** %m1
  %j26 = load i32* %j
  %tmp27 = add i32 %j26, 1
  store i32 %tmp27, i32* %j
  br label %while7

then:                                             ; preds = %while_body8
  %printf15 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str23,
i32 0, i32 0))
  br label %merge

else:                                             ; preds = %while_body8
  %i16 = load i32* %i
  %len_minus_117 = load i32* %len_minus_1
  %tmp18 = icmp slt i32 %i16, %len_minus_117
  br i1 %tmp18, label %then20, label %else22

merge19:                                          ; preds = %else22, %then20
  br label %merge

then20:                                           ; preds = %else
  %printf21 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @.str24,
i32 0, i32 0))
  br label %merge19

else22:                                           ; preds = %else
  br label %merge19

merge31:                                          ; preds = %while7
  %i32 = load i32* %i
  %tmp33 = add i32 %i32, 1
  store i32 %tmp33, i32* %i
  br label %while

merge37:                                          ; preds = %while
  %printf38 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str25,
i32 0, i32 0))
  ret void
```

```
}

define void @print_rowt([3 x i32]* %r, i32 %len) {
entry:
  %r1 = alloca [3 x i32]*
  store [3 x i32]* %r, [3 x i32]** %r1
  %len2 = alloca i32
  store i32 %len, i32* %len2
  %i = alloca i32
  %len_minus_1 = alloca i32
  %_i = alloca i32
  %len3 = load i32* %len2
  %tmp = sub i32 %len3, 1
  store i32 %tmp, i32* %len_minus_1
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str29, i32
0, i32 0))
  store i32 0, i32* %i
  br label %while

while:                                            ; preds = %merge, %entry
  %i15 = load i32* %i
  %len16 = load i32* %len2
  %tmp17 = icmp slt i32 %i15, %len16
  br i1 %tmp17, label %while_body, label %merge18

while_body:                                       ; preds = %while
  %r4 = load [3 x i32]** %r1
  %r5 = load [3 x i32]* %r4
  call void @print_tuple([3 x i32] %r5)
  %i6 = load i32* %i
  %len_minus_17 = load i32* %len_minus_1
  %tmp8 = icmp slt i32 %i6, %len_minus_17
  br i1 %tmp8, label %then, label %else

merge:                                            ; preds = %else, %then
  %r10 = getelementptr inbounds [3 x i32]** %r1, i32 0
  %r11 = load [3 x i32]** %r10
  %r12 = getelementptr inbounds [3 x i32]* %r11, i32 1
  store [3 x i32]* %r12, [3 x i32]** %r1
  %i13 = load i32* %i
  %tmp14 = add i32 %i13, 1
  store i32 %tmp14, i32* %i
  br label %while

then:                                             ; preds = %while_body
  %printf9 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str30, i32
0, i32 0))
  br label %merge
```

```
else:                                             ; preds = %while_body
  br label %merge

merge18:                                          ; preds = %while
  %printf19 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str31,
i32 0, i32 0))
  ret void
}

define void @print_rowf(double* %r, i32 %len) {
entry:
  %r1 = alloca double*
  store double* %r, double** %r1
  %len2 = alloca i32
  store i32 %len, i32* %len2
  %i = alloca i32
  %len_minus_1 = alloca i32
  %_i = alloca i32
  %len3 = load i32* %len2
  %tmp = sub i32 %len3, 1
  store i32 %tmp, i32* %len_minus_1
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str35, i32
0, i32 0))
  store i32 0, i32* %i
  br label %while

while:                                            ; preds = %merge, %entry
  %i16 = load i32* %i
  %len17 = load i32* %len2
  %tmp18 = icmp slt i32 %i16, %len17
  br i1 %tmp18, label %while_body, label %merge19

while_body:                                       ; preds = %while
  %r4 = load double** %r1
  %r5 = load double* %r4
  %printf6 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @fmt33, i32
0, i32 0), double %r5)
  %i7 = load i32* %i
  %len_minus_18 = load i32* %len_minus_1
  %tmp9 = icmp slt i32 %i7, %len_minus_18
  br i1 %tmp9, label %then, label %else

merge:                                            ; preds = %else, %then
  %r11 = getelementptr inbounds double** %r1, i32 0
  %r12 = load double** %r11
  %r13 = getelementptr inbounds double* %r12, i32 1
  store double* %r13, double** %r1
  %i14 = load i32* %i
  %tmp15 = add i32 %i14, 1
```

```
  store i32 %tmp15, i32* %i
  br label %while
then:                                             ; preds = %while_body
  %printf10 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str36,
i32 0, i32 0))
  br label %merge

else:                                             ; preds = %while_body
  br label %merge

merge19:                                          ; preds = %while
  %printf20 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str37,
i32 0, i32 0))
  ret void
}

define void @print_rowi(i32* %r, i32 %len) {
entry:
  %r1 = alloca i32*
  store i32* %r, i32** %r1
  %len2 = alloca i32
  store i32 %len, i32* %len2
  %i = alloca i32
  %len_minus_1 = alloca i32
  %_i = alloca i32
  %len3 = load i32* %len2
  %tmp = sub i32 %len3, 1
  store i32 %tmp, i32* %len_minus_1
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str41, i32
0, i32 0))
  store i32 0, i32* %i
  br label %while

while:                                            ; preds = %merge, %entry
  %i16 = load i32* %i
  %len17 = load i32* %len2
  %tmp18 = icmp slt i32 %i16, %len17
  br i1 %tmp18, label %while_body, label %merge19

while_body:                                       ; preds = %while
  %r4 = load i32** %r1
  %r5 = load i32* %r4
  %printf6 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @fmt38, i32
0, i32 0), i32 %r5)
  %i7 = load i32* %i
  %len_minus_18 = load i32* %len_minus_1
  %tmp9 = icmp slt i32 %i7, %len_minus_18
  br i1 %tmp9, label %then, label %else
```

```
merge:                                            ; preds = %else, %then
  %r11 = getelementptr inbounds i32** %r1, i32 0
  %r12 = load i32** %r11
  %r13 = getelementptr inbounds i32* %r12, i32 1
  store i32* %r13, i32** %r1
  %i14 = load i32* %i
  %tmp15 = add i32 %i14, 1
  store i32 %tmp15, i32* %i
  br label %while

then:                                             ; preds = %while_body
  %printf10 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str42,
i32 0, i32 0))
  br label %merge

else:                                             ; preds = %while_body
  br label %merge

merge19:                                          ; preds = %while
  %printf20 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str43,
i32 0, i32 0))
  ret void
}

define void @print_tuple([3 x i32] %t) {
entry:
  %t1 = alloca [3 x i32]
  store [3 x i32] %t, [3 x i32]* %t1
  %i = alloca i32
  %_i = alloca i32
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str47, i32
0, i32 0))
  store i32 0, i32* %i
  br label %while

while:                                            ; preds = %merge, %entry
  %i10 = load i32* %i
  %tmp11 = icmp slt i32 %i10, 3
  br i1 %tmp11, label %while_body, label %merge12

while_body:                                       ; preds = %while
  %i2 = load i32* %i
  %t3 = getelementptr [3 x i32]* %t1, i32 0, i32 %i2
  %t4 = load i32* %t3
  %printf5 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @fmt44, i32
0, i32 0), i32 %t4)
  %i6 = load i32* %i
  %tmp = icmp slt i32 %i6, 2
  br i1 %tmp, label %then, label %else
```

```
merge:                                            ; preds = %else, %then
  %i8 = load i32* %i
  %tmp9 = add i32 %i8, 1
  store i32 %tmp9, i32* %i
  br label %while

then:                                             ; preds = %while_body
  %printf7 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str48, i32
0, i32 0))
  br label %merge

else:                                             ; preds = %while_body
  br label %merge

merge12:                                          ; preds = %while
  %printf13 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str49,
i32 0, i32 0))
  ret void
}

define i32 @main() {
entry:
  %int_row = alloca [4 x i32]
  %int_matrix = alloca [2 x [4 x i32]]
  %float_row = alloca [3 x double]
  %float_matrix = alloca [2 x [3 x double]]
  %t = alloca [3 x i32]
  %tuple_row = alloca [2 x [3 x i32]]
  %tuple_matrix = alloca [2 x [2 x [3 x i32]]]
  %_i = alloca i32
  store [4 x i32] [i32 1, i32 2, i32 3, i32 4], [4 x i32]* %int_row
  store [2 x [4 x i32]] [[4 x i32] [i32 1, i32 2, i32 3, i32 4], [4 x i32] [i32 5, i32 6,
i32 7, i32 8]], [2 x [4 x i32]]* %int_matrix
  store [3 x double] [double 1.100000e+00, double 2.200000e+00, double 3.300000e+00], [3 x
double]* %float_row
  store [2 x [3 x double]] [[3 x double] [double 1.100000e+00, double 2.200000e+00, double
3.300000e+00], [3 x double] [double 4.400000e+00, double 5.500000e+00, double
6.600000e+00]], [2 x [3 x double]]* %float_matrix
  store [3 x i32] [i32 1, i32 2, i32 3], [3 x i32]* %t
  store [2 x [3 x i32]] [[3 x i32] [i32 1, i32 2, i32 3], [3 x i32] [i32 3, i32 4, i32 5]],
[2 x [3 x i32]]* %tuple_row
  store [2 x [2 x [3 x i32]]] [[2 x [3 x i32]] [[3 x i32] [i32 1, i32 2, i32 3], [3 x i32]
[i32 4, i32 5, i32 6]], [2 x [3 x i32]] [[3 x i32] [i32 7, i32 8, i32 9], [3 x i32] [i32 10,
i32 11, i32 12]]], [2 x [2 x [3 x i32]]]* %tuple_matrix
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([8 x i8]* @.str53, i32
0, i32 0))
  %int_row1 = getelementptr [4 x i32]* %int_row, i32 0
  %int_row2 = load [4 x i32]* %int_row1
```

```
  %int_row3 = getelementptr inbounds [4 x i32]* %int_row, i32 0, i32 0
  call void @print_rowi(i32* %int_row3, i32 4)
  %printf4 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([2 x i8]* @.str54, i32
0, i32 0))
  %int_matrix5 = getelementptr [2 x [4 x i32]]* %int_matrix, i32 0, i32 0
  %int_matrix6 = load [4 x i32]* %int_matrix5
  %int_matrix7 = getelementptr [2 x [4 x i32]]* %int_matrix, i32 0
  %int_matrix8 = load [2 x [4 x i32]]* %int_matrix7
  %int_matrix9 = getelementptr inbounds [2 x [4 x i32]]* %int_matrix, i32 0, i32 0, i32 0
  call void @print_matrixi(i32* %int_matrix9, i32 2, i32 4)
  %printf10 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([2 x i8]* @.str55,
i32 0, i32 0))
  %printf11 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([10 x i8]* @.str56,
i32 0, i32 0))
  %float_row12 = getelementptr [3 x double]* %float_row, i32 0
  %float_row13 = load [3 x double]* %float_row12
  %float_row14 = getelementptr inbounds [3 x double]* %float_row, i32 0, i32 0
  call void @print_rowf(double* %float_row14, i32 3)
  %printf15 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([2 x i8]* @.str57,
i32 0, i32 0))
  %float_matrix16 = getelementptr [2 x [3 x double]]* %float_matrix, i32 0, i32 0
  %float_matrix17 = load [3 x double]* %float_matrix16
  %float_matrix18 = getelementptr [2 x [3 x double]]* %float_matrix, i32 0
  %float_matrix19 = load [2 x [3 x double]]* %float_matrix18
  %float_matrix20 = getelementptr inbounds [2 x [3 x double]]* %float_matrix, i32 0, i32 0,
i32 0
  call void @print_matrixf(double* %float_matrix20, i32 2, i32 3)
  %printf21 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([2 x i8]* @.str58,
i32 0, i32 0))
  %printf22 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([10 x i8]* @.str59,
i32 0, i32 0))
  %t23 = load [3 x i32]* %t
  call void @print_tuple([3 x i32] %t23)
  %printf24 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([2 x i8]* @.str60,
i32 0, i32 0))
  %tuple_row25 = getelementptr [2 x [3 x i32]]* %tuple_row, i32 0
  %tuple_row26 = load [2 x [3 x i32]]* %tuple_row25
  %tuple_row27 = getelementptr inbounds [2 x [3 x i32]]* %tuple_row, i32 0, i32 0
  call void @print_rowt([3 x i32]* %tuple_row27, i32 2)
  %printf28 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([2 x i8]* @.str61,
i32 0, i32 0))
  %tuple_matrix29 = getelementptr [2 x [2 x [3 x i32]]]* %tuple_matrix, i32 0, i32 0
  %tuple_matrix30 = load [2 x [3 x i32]]* %tuple_matrix29
  %tuple_matrix31 = getelementptr [2 x [2 x [3 x i32]]]* %tuple_matrix, i32 0
  %tuple_matrix32 = load [2 x [2 x [3 x i32]]]* %tuple_matrix31
  %tuple_matrix33 = getelementptr inbounds [2 x [2 x [3 x i32]]]* %tuple_matrix, i32 0, i32
0, i32 0
  call void @print_matrixt([3 x i32]* %tuple_matrix33, i32 2, i32 2)
```

```
  %printf34 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([2 x i8]* @.str62,
i32 0, i32 0))
  ret i32 0
}
```

b. Test Example 2

This example displays how to declare a matrix of a 2x2 size, declare a pointer to it, access some elements and dereference the pointer to return a value. This specific example just deals with ints, but the same can be done with floats and tuples.

b.1 Example 2 in Native Language

`pointer_test.JSTEM`

```
def int main(){
  int[2][2] y;
  int[[]] p;
  int q;

  y[0][0] = 1;
  y[0][1] = 2;
  y[1][0] = 3;
  y[1][1] = 4;

  p = $$y;
  p = ~~p;
  p = ~~p;
  q = #p;

  print_matrixi($$y, 2, 2);
  prints("\n");
  print(q);

  return 0;
}
```

`pointer_test.JSTEM`

```
{ 1, 2
  3, 4 }
3
```

b.2 Example 2 in Target Language

```
define i32 @main() {
entry:
  %y = alloca [2 x [2 x i32]]
  %p = alloca i32*
  %q = alloca i32
  %_i = alloca i32
  %y1 = getelementptr [2 x [2 x i32]]* %y, i32 0, i32 0, i32 0
  store i32 1, i32* %y1
  %y2 = getelementptr [2 x [2 x i32]]* %y, i32 0, i32 0, i32 1
  store i32 2, i32* %y2
  %y3 = getelementptr [2 x [2 x i32]]* %y, i32 0, i32 1, i32 0
  store i32 3, i32* %y3
  %y4 = getelementptr [2 x [2 x i32]]* %y, i32 0, i32 1, i32 1
  store i32 4, i32* %y4
  %y5 = getelementptr inbounds [2 x [2 x i32]]* %y, i32 0, i32 0, i32 0
  store i32* %y5, i32** %p
  %p6 = getelementptr inbounds i32** %p, i32 0
  %p7 = load i32** %p6
  %p8 = getelementptr inbounds i32* %p7, i32 1
  store i32* %p8, i32** %p
  %p9 = getelementptr inbounds i32** %p, i32 0
  %p10 = load i32** %p9
  %p11 = getelementptr inbounds i32* %p10, i32 1
  store i32* %p11, i32** %p
  %p12 = load i32** %p
  %p13 = load i32* %p12
  store i32 %p13, i32* %q
  %y14 = getelementptr inbounds [2 x [2 x i32]]* %y, i32 0, i32 0, i32 0
  call void @print_matrixi(i32* %y14, i32 2, i32 2)
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([2 x i8]* @.str53, i32
0, i32 0))
  %q15 = load i32* %q
  %printf16 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @fmt50, i32
0, i32 0), i32 %q15)
  ret i32 0
}
```

c. Test Cases

The tests/compiler_tests directory contains all of the test files. The .JSTEM files that start with “test-” are the successful tests. The expected results of those tests are in the file with same name but with the file extension “.out”. The .JSTEM files that start with “fail-” are the fail tests. The expected output of the tests are in the file with the same name but with the file extension “.err”. We created a comprehensive test suite to test all aspects of our language. The test suite includes fail and successful files which test assignment, for and while loops, if statements, arithmetic

operations, global variables, function calling, tuple, row, and matrix declaration, access, operations, and printing.

7. Lessons Learned

a. Overall Challenges

Some of the challenges that the team faced included how the language changed. While the team knew they wanted to make a matrix manipulation language, there was some debate on whether or not some features should still be included as the team faced troubles implementing certain features. Initially we were overly optimistic in wanting to develop a language that would manipulate and alter images. However, we ended up narrowing this down by reading in more of a .csv file format instead of a .ppm file. In this representation, integer values were separated by commas, and rows were separated by newlines. Once read into a matrix, we could manipulate it using our matrix operations and then write the new matrix back into a similar file type.

b. Tessa Hurr

I learned a lot about the parts that need to come together to form a compiler. Through struggling with this project, I realized how hard it is to make a good, comprehensive language that is easy to use. Something I struggled with was figuring out how to make progress when I felt like we weren't given too much information because there is very little documentation online that is helpful. I also realized that I shouldn't be afraid to ask for more help and keep asking TA's for clarification on areas that I am unclear about because this is the majority of the help that we could get throughout doing the project. Also, while much of this project was helpful to start together, I also realized how important it was to keep working and struggling with implementing features outside of the times that our group met.

c. Julia Troxell

Initially, I struggled most with being mentally overwhelmed with the sheer amount of unfamiliar code from microc/past projects. I learned that simply staring at this code did not help me learn what it was doing. Instead, once I started trying to write my own functions, the debugging process really helped me understand what was happening in the code and how the compiler all fit together.

d. Michelle Lu

As we started to add more features and build on top of what we had previously implemented, I began to see the importance of writing good and in-depth fail and success tests. There were some

bugs in features that we did not catch until over a week after we created them. The tests regarding these features passed so we thought they were fully functioning. However, we forgot to add tests that checked all types. In addition, I saw that throwing specific failure exceptions for the failure became extremely helpful when trying to debug in the future. Writing tests to cover all cases might be tedious but is definitely necessary to ensure that your functions are completely working.

e. Emily Song

I learned how important it is to understand the fundamentals, which in this case entails programming in OCaml and understanding the structure of a compiler. It makes debugging much less difficult when you know, on a very specific level, what each component is supposed to do, and how the code helps accomplish that task. In addition, I learned how important it is to set concrete goals and deadlines, because this project has such a large scope, and breaking it up into smaller, more manageable components definitely helped us to stay on track to finish.

f. Sam Stultz

Overall, learnt how exactly a compiler works and the logic the logic that is required for constructing a compiler. I think the most interesting thing I learnt was about the importance of language unambiguity. Starting out we all had the idea for a simple syntax, but throughout the project we ended up moving further away from it as we kept introducing new tokens as we implemented new data structures such as rows and tuples. Also, while implementing files and string manipulation functions I learnt how important memory allocation was. I didn't realize that an understanding of C was so important for me to understand what I was implementing.

8. Appendix

All group members contributed to the following files as the implementation plan was to pick a feature and implement it end-to-end. However, the previous statement holds true with the exception of the Makefile (written by Sam Stultz) and stdlib.JSTEM (written by Julia Troxell) which has been noted in the section for each of these files below.

a.

`Scanner.mll`

```
(* Ocamllex scanner for MicroC *)

{ open Parser

  let un_esc s =
       Scanf.sscanf ("\"" ^ s ^ "\"") "%S%!" (fun x -> x)
}
```

```
let digits = ['0'-'9']
let alphabet = ['a'-'z' 'A'-'Z']
let dig_or_alpha = digits| alphabet| '_'
let id = alphabet dig_or_alpha*

let esc_char = '\\' ['\\' ''' '"' 'n' 'r' 't']
let ascii_sym = ([' '-'!' '#'-'[' ']'-'~'])
let char = '"' ( ascii_sym | digits ) '"'
let string = '"' ( (ascii_sym | esc_char)* as s ) '"'

rule token = parse
  [' ' '\t' '\r' '\n'] { token lexbuf } (* Whitespace *)
| "/*"     { comment lexbuf }           (* Comments *)
| '('      { LPAREN }
| ')'      { RPAREN }
| '{'      { LBRACE }
| '}'      { RBRACE }
| '['      { LSQBRACE }
| ']'      { RSQBRACE }
| "(%"     { LPERCENT }
| "%)"     { RPERCENT }
| "{%"     { LMPERCENT }
| "%}"     { RMPERCENT }
| ';'      { SEMI }
| ':'      { COLON }
| ','      { COMMA }
| '|'      { BAR }
| '+'      { PLUS }
| '-'      { MINUS }
| '*'      { TIMES }
| '/'      { DIVIDE }
| "@="     { ATASSIGN }
| '='      { ASSIGN }
| "=="     { EQ }
| "!="     { NEQ }
| '<'      { LT }
| "<="     { LEQ }
| ">"      { GT }
| ">="     { GEQ }
| "&&"     { AND }
| "||"     { OR }
| "!"      { NOT }
| "if"     { IF }
| "else"   { ELSE }
| "for"    { FOR }
| "while"  { WHILE }
| "return" { RETURN }
| "int"    { INT }
```

```
| "bool"   { BOOL }
| "float"  { FLOAT }
| "void"   { VOID }
| "length" { LENGTH }
| "width"  { WIDTH }
| "type"   { TYPE }
| "tuple"  { TUPLE }
| "def"    { DEF }
| "in"     { IN }
| "True"   { TRUE }
| "False"  { FALSE }
| "$"       { DOLLAR }
| "#"       { OCTOTHORP }
| "new"    { NEW }
| "String" { STRING }
| "File"   { STRING }
| "~"      { SQUIGLY }
| "NULL"   { NULL}
| ['0'-'9']+'.'['0'-'9']+ as lxm { FLOAT_LIT(float_of_string lxm) }
| '.'      { DOT }
| ['0'-'9']+ as lxm { INT_LIT(int_of_string lxm) }
| id as lxm { ID(lxm) }
| string  { STRING_LIT(un_esc s) }
| eof { EOF }
| _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }


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

b.

parser.mly

```
%{
open Ast
%}

%token SEMI LPAREN RPAREN LBRACE RBRACE COMMA BAR COLON LSQBRACE RSQBRACE LPERCENT RPERCENT
LMPERCENT RMPERCENT
%token PLUS MINUS TIMES DIVIDE ASSIGN NOT ATASSIGN NULL
%token EQ NEQ LT LEQ GT GEQ AND OR NEW
%token TRUE FALSE
%token RETURN IF ELSE FOR WHILE INT BOOL VOID FLOAT TUPLE STRING
%token OCTOTHORP DOLLAR SQUIGLY

%token <int> INT_LIT
%token <string> STRING_LIT
```

```
%token <string> ID
%token <float> FLOAT_LIT

%token DEF
%token IN
%token DOT
%token LENGTH WIDTH TYPE
%token EOF

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

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

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

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

```
matrix_typ:
  primitive LSQBRACE INT_LIT RSQBRACE LSQBRACE INT_LIT RSQBRACE { MatrixTyp($1, $3, $6) }

row_typ:
    primitive LSQBRACE INT_LIT RSQBRACE { RowTyp($1, $3) }

tuple_typ:
  typ LPERCENT INT_LIT RPERCENT { TupleTyp($1, $3) }

row_pointer_typ:
  primitive LSQBRACE RSQBRACE { RowPointer($1) }

matrix_pointer_typ:
  primitive LSQBRACE LSQBRACE RSQBRACE RSQBRACE { MatrixPointer($1) }

typ:
  primitive { $1 }
  | matrix_typ { $1 }
  | row_typ { $1 }

primitive:
       INT { Int }
  | BOOL { Bool }
  | VOID { Void }
  | FLOAT { Float }
  | STRING { String }
  | tuple_typ { $1 }
  | row_pointer_typ { $1 }
  | matrix_pointer_typ { $1 }

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

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

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

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

```
  | FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN stmt
     { For($3, $5, $7, $9) }
  | FOR LPAREN ID IN ID RPAREN stmt { MFor($3, $5, $7) }
  | WHILE LPAREN expr RPAREN stmt { While($3, $5) }

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

expr:
    literals          { $1 }
  | ID                        { Id($1) }
  | expr PLUS   expr { Binop($1, Add,   $3) }
  | expr MINUS  expr { Binop($1, Sub,   $3) }
  | expr TIMES  expr { Binop($1, Mult,  $3) }
  | expr DIVIDE expr { Binop($1, Div,   $3) }
  | expr EQ     expr { Binop($1, Equal, $3) }
  | expr NEQ    expr { Binop($1, Neq,   $3) }
  | expr LT     expr { Binop($1, Less,  $3) }
  | expr LEQ    expr { Binop($1, Leq,   $3) }
  | expr GT     expr { Binop($1, Greater, $3) }
  | expr GEQ    expr { Binop($1, Geq,   $3) }
  | expr AND    expr { Binop($1, And,   $3) }
  | expr OR     expr { Binop($1, Or,    $3) }
  | MINUS expr %prec NEG { Unop(Neg, $2) }
  | NOT expr         { Unop(Not, $2) }
  | expr ASSIGN expr   { Assign($1, $3) }
  | ID LPAREN actuals_opt RPAREN { Call($1, $3) }
  | LPAREN expr RPAREN { $2 }
  | ID ATASSIGN NEW LSQBRACE expr RSQBRACE { Init($1,$5) }
  | DOLLAR ID { RowReference($2) }
  | DOLLAR DOLLAR ID { MatrixReference($3) }
  | OCTOTHORP ID { Dereference($2) }
  | SQUIGLY SQUIGLY ID { PointerIncrement($3) }
  | ID LSQBRACE expr RSQBRACE { RowAccess($1, $3) }
  | ID LPERCENT expr RPERCENT { TupleAccess($1, $3) }
  | ID LSQBRACE expr RSQBRACE LSQBRACE expr RSQBRACE { MatrixAccess($1, $3, $6) }
  | ID LSQBRACE expr RSQBRACE LSQBRACE COLON RSQBRACE { MRowAccess($1, $3) }
  | ID DOT LENGTH { Length($1) }
  | ID DOT WIDTH { Width($1) }
  | ID DOT TYPE { Type($1) }
  | NULL LPAREN expr RPAREN { Null($3) }

primitives:
       INT_LIT      { IntLit($1) }
  | FLOAT_LIT  { FloatLit($1) }
  | STRING_LIT { StringLit($1) }
  | TRUE       { BoolLit(true) }
  | FALSE      { BoolLit(false) }
```

```
literals:
       primitives { $1 }
  |    tuple_literal  { $1 }
  |    LSQBRACE primitive_rowlit RSQBRACE { RowLit(List.rev $2) }
  | LSQBRACE tuple_rowlit RSQBRACE { RowLit(List.rev $2) }
  | LMPERCENT primitive_matrixlit RMPERCENT { MatrixLit(List.rev $2) }
  | LMPERCENT tuple_matrixlit RMPERCENT { MatrixLit(List.rev $2) }


primitive_rowlit:
       primitives { [$1] }
  | primitive_rowlit COMMA primitives { $3 :: $1 }

tuple_rowlit:
       tuple_literal { [$1] }
  | tuple_rowlit COMMA tuple_literal { $3 :: $1 }

tuple_matrixlit:
       tuple_rowlit { [$1] }
  | tuple_matrixlit BAR tuple_rowlit { $3 :: $1 }

primitive_matrixlit:
       primitive_rowlit { [$1] }
  | primitive_matrixlit BAR primitive_rowlit { $3 :: $1 }

tuple_literal:
  LPERCENT array_literal RPERCENT { TupleLit(List.rev $2) }

array_literal:
  primitives { [$1] }
  | array_literal COMMA primitives { $3 :: $1 }

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

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

c.

ast.ml

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

type uop = Neg | Not
```

```
type typ =
        Int
  | Float
  | String
  | Bool
  | Void
  | Tuple
  | Row
  | MatrixTyp of typ * int * int
  | RowTyp of typ * int
  | TupleTyp of typ * int
  | RowPointer of typ
  | MatrixPointer of typ

type bind = typ * string

type expr = IntLit of int
  | Id of string
  | StringLit of string
  | BoolLit of bool
  | FloatLit of float
  | TupleLit of expr list
  | MatrixLit of expr list list
  | RowLit of expr list
  | Init of string * expr
  | Binop of expr * op * expr
  | Unop of uop * expr
  | Assign of expr * expr
  | Call of string * expr list
  | RowAccess of string * expr
  | TupleAccess of string * expr
  | MatrixAccess of string * expr * expr
  | MRowAccess of string * expr
  | Noexpr
  | MatrixReference of string
  | PointerIncrement of string
  | Dereference of string
  | RowReference of string
  | Length of string
  | Width of string
  | Type of string
  | Null of expr

type stmt = Block of stmt list
  | Expr of expr
  | If of expr * stmt * stmt
  | For of expr * expr * expr * stmt
  | MFor of string * string * stmt
```

```
  | While of expr * stmt
  | Return of expr

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

type program = bind list * func_decl list

(* Pretty-printing functions *)

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

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

let string_of_tuple t =
  let rec string_of_tuple_literal = function
    [] -> ")"
  | [hd] -> (match hd with
      IntLit(i)-> string_of_int i
    | _ -> raise( Failure("Illegal expression in tuple primitive") )) ^
string_of_tuple_literal []
  | hd :: tl -> (match hd with
         IntLit(i) -> string_of_int i ^ ", "
    | _ -> raise( Failure("Illegal expression in tuple primitive") )) ^
string_of_tuple_literal tl
in
"(" ^ string_of_tuple_literal t

let string_of_row r =
  let rec string_of_row_literal = function
```

```
       [] -> "]"
  | [hd] -> (match hd with
         IntLit(i) -> string_of_int i
       | FloatLit(f) -> string_of_float f
       | TupleLit(t) -> string_of_tuple t
       | _ -> raise( Failure("Illegal expression in row primitive") )) ^
string_of_row_literal []
  | hd :: tl -> (match hd with
         IntLit(i) -> string_of_int i ^ ", "
       | FloatLit(f) -> string_of_float f ^ ", "
       | TupleLit(t) -> string_of_tuple t ^ ", "
       | _ -> raise( Failure("Illegal expression in row primitive") )) ^
string_of_row_literal tl
in
"[" ^ string_of_row_literal r

let string_of_matrix m =
 let rec string_of_matrix_literal = function
      [] -> "}"
    | [hd] -> (match hd with
                RowLit(r) -> string_of_row r
              | _ -> raise( Failure("Illegal expression in matrix primitive") )) ^
string_of_matrix_literal []
    | hd::tl -> (match hd with
                    RowLit(r) -> string_of_row r ^ ", "
                  | _ -> raise( Failure("Illegal expression in matrix primitive") )) ^
string_of_matrix_literal tl
  in
  "{" ^ string_of_matrix_literal m

let rec string_of_expr = function
    IntLit(i) -> string_of_int i
  | BoolLit(b) -> string_of_bool b
  | StringLit(s) -> s
  | Id(i) -> i
  | FloatLit(f) -> string_of_float f
  | MatrixLit(_)-> "matrix literal"
  | TupleLit(t) -> string_of_tuple t
  | RowLit(r) -> string_of_row r
  | Binop(e1, o, e2) ->
      string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
  | Unop(o, e) -> string_of_uop o ^ string_of_expr e
  | Assign(v, e) -> string_of_expr v ^ " = " ^ string_of_expr e
  | Call(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | RowAccess(r, e) -> r ^ "[" ^ string_of_expr e ^ "]"
  | TupleAccess(t, e) -> t ^ "(%" ^ string_of_expr e ^ "%)"
  | MatrixAccess(m, e1, e2) -> m ^ "[" ^ string_of_expr e1 ^ "][" ^ string_of_expr e2 ^ "]"
  | MRowAccess(r, e) -> r ^ "[" ^ string_of_expr e ^ "][:]"
```

```
  | Init(v,e) -> v ^ "=" ^ "new" ^ "[" ^ string_of_expr e ^ "]"
  | Noexpr -> ""
  | RowReference(s) -> "$" ^ s
  | MatrixReference(s) -> "$$" ^ s
  | Dereference(s) -> "#" ^ s
  | Length(s) -> s ^ ".length"
  | Width(s) -> s ^ ".width"
  | Type(s) -> s ^ ".type"
  | PointerIncrement(s) -> "~~" ^ s
  | Null(e) -> "null" ^ "(" ^ string_of_expr e ^ ")"


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
  | MFor(s1, s2, s) -> "for (" ^ s1 ^ " in " ^ s2 ^ ")\n" ^ string_of_stmt s
  | While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^ string_of_stmt s

let rec string_of_typ = function
    Int -> "int"
  | Bool -> "bool"
  | Void -> "void"
  | Float -> "float"
  | String -> "string"
  | MatrixTyp(t, l1, l2) -> (match t with
                        Int -> "int" ^ "[" ^ string_of_int l1 ^ "][" ^ string_of_int l2 ^
"]"
                      | Float -> "float" ^ "[" ^ string_of_int l1 ^ "][" ^ string_of_int l2
^ "]"
                      | TupleTyp(x, l) -> (match x with
                                            Int -> "int" ^ "(%" ^ string_of_int l ^ "%)[" ^
string_of_int l1 ^ "][" ^ string_of_int l2 ^ "]"
                                                                                | _ -> raise(
Failure("Illegal expression in tuple primitive") ))
                                       | _ -> raise( Failure("Illegal expression in matrix
primitive")))
  | TupleTyp(x, l) -> (match x with
                      Int -> "int" ^ "(%" ^ string_of_int l ^ "%)"
                                     | _ -> raise( Failure("Illegal expression in tuple
primitive")))
  | RowTyp(r, l1) -> (match r with
```

```
                                        Int -> "int" ^ "[" ^ string_of_int l1 ^ "]"
                    | Float -> "float" ^ "[" ^ string_of_int l1 ^ "]"
                    | TupleTyp(x, l) -> (match x with
                                        Int -> "int" ^ "(%" ^ string_of_int l ^ "%)[" ^
string_of_int l1 ^ "]"
                                                                        | _ -> raise(
Failure("Illegal expression in tuple primitive") ))
                                    | _ -> raise( Failure("Illegal expression in row
primitive")))
  | RowPointer(t) -> string_of_typ t ^ "[]"
  | MatrixPointer(t) -> string_of_typ t ^ "[[]]"
  | _ -> raise( Failure("Illegal expression in string_of_typ"))


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

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

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

d.

semant.ml

```
open Ast

module StringMap = Map.Make(String)

(* Semantic checking of a program. Returns void if successful,
   throws an exception if something is wrong.

   Check each global variable, then check each function *)

let check (globals, functions) =

  (* Raise an exception if the given list has a duplicate *)
  let report_duplicate exceptf list =
    let rec helper = function
        n1 :: n2 :: _ when n1 = n2 -> raise (Failure (exceptf n1))
      | _ :: t -> helper t
```

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

  (* Raise an exception if a given binding is to a void type *)
  let check_not_void exceptf = function
      (Void, n) -> raise (Failure (exceptf n))
    | _ -> ()
  in

  let check_assign lvaluet rvaluet err =
    match (lvaluet, rvaluet) with
      (Int, Int) -> lvaluet
    | (Float, Float) -> lvaluet
    | (String, String) -> lvaluet
    | (Bool, Bool) -> lvaluet
    | (Void, Void) -> lvaluet
    | (TupleTyp(Int, l1), TupleTyp(Int, l2)) -> if l1 == l2 then lvaluet else if l1 == 0
then lvaluet else raise err
    | (RowTyp(Int, l1), RowTyp(Int, l2)) -> if l1 == l2 then lvaluet else if l1 == 0 then
lvaluet else raise err
    | (RowTyp(Float, l1), RowTyp(Float, l2)) -> if l1 == l2 then lvaluet else if l1 == 0
then lvaluet else raise err
    | (RowTyp(TupleTyp(Int, d1), l1), RowTyp(TupleTyp(Int, d2), l2)) -> if d1 == d2 && l1 ==
l2 then lvaluet else if l1 == 0 then lvaluet else raise err
    | (MatrixTyp(Int, r1, c1), MatrixTyp(Int, r2, c2)) -> if r1 == r2 && c1 == c2 then
lvaluet else raise err
    | (MatrixTyp(Float, r1, c1), MatrixTyp(Float, r2, c2)) -> if r1 == r2 && c1 == c2 then
lvaluet else raise err
    | (MatrixTyp(TupleTyp(Int, d1), r1, c1), MatrixTyp(TupleTyp(Int, d2), r2, c2)) -> if d1
== d2 && r1 == r2 && c1 == c2 then lvaluet else raise err
    | (MatrixPointer(Int), MatrixPointer(Int)) -> lvaluet
    | (MatrixPointer(Float), MatrixPointer(Float)) -> lvaluet
    | (MatrixPointer(TupleTyp(Int, 3)), MatrixPointer(TupleTyp(Int, 3))) -> lvaluet
    | (RowPointer(Int), RowPointer(Int)) -> lvaluet
    | (RowPointer(Float), RowPointer(Float)) -> lvaluet
    | (RowPointer(TupleTyp(Int, 3)), RowPointer(TupleTyp(Int, 3))) -> lvaluet
    | _ -> raise err
  in

  (**** Checking Global Variables ****)

  List.iter (check_not_void (fun n -> "illegal void global " ^ n)) globals;

  report_duplicate (fun n -> "duplicate global " ^ n) (List.map snd globals);

    (**** Checking Functions ****)

  if List.mem "print" (List.map (fun fd -> fd.fname) functions)
```

```
  then raise (Failure ("function print may not be defined")) else ();

  if List.mem "prints" (List.map (fun fd -> fd.fname) functions)
  then raise (Failure ("function print may not be defined")) else ();

  if List.mem "printb" (List.map (fun fd -> fd.fname) functions)
  then raise (Failure ("function print may not be defined")) else ();

  if List.mem "printf" (List.map (fun fd -> fd.fname) functions)
  then raise (Failure ("function print may not be defined")) else ();

  report_duplicate (fun n -> "duplicate function " ^ n)
  (List.map (fun fd -> fd.fname) functions);


  let built_in_decls = StringMap.add "print"
  { typ = Void; fname = "print"; formals = [(Int, "x")];
    locals = []; body = [] } (StringMap.add "printf"
  { typ = Void; fname = "printf"; formals = [(Float, "x")];
    locals = []; body = [] } (StringMap.add "prints"
  { typ = Void; fname = "prints"; formals = [(String, "x")];
    locals = []; body = [] } (StringMap.add "printfs"
  { typ = Void; fname = "printfs"; formals = [(String, "x")];
    locals = []; body = [] } (StringMap.add "printb"
  { typ = Void; fname = "printb"; formals = [(Bool, "x")];
    locals = []; body = [] } (StringMap.add "open"
  { typ = String; fname = "open"; formals = [(String, "x"); (String,"y")];
    locals = []; body = [] } (StringMap.add "read"
  { typ = String; fname = "read"; formals = [(String,"w"); (Int, "x"); (Int, "y"); (String,
"z")];
    locals = []; body = [] } (StringMap.add "write"
  { typ = Int; fname = "write"; formals = [(String, "w"); (Int, "x"); (Int, "y"); (String,
"z")];
    locals = []; body = [] } (StringMap.add "close"
  { typ = Void; fname = "close"; formals = [(String, "x")];
    locals = []; body = [] } (StringMap.add "fget"
  { typ = String; fname = "fget"; formals = [(String, "x"); (Int, "y"); (String, "z")];
    locals = []; body = [] } (StringMap.add "atoi"
  { typ = Int ; fname = "atoi"; formals = [(String, "x")];
    locals = []; body = [] } (StringMap.add "itos"
  { typ = Void ; fname = "itos"; formals = [(String, "x"); (String,"y"); (Int, "z")];
    locals = []; body = [] } (StringMap.add "splitstr"
  { typ = String ; fname = "splitstr"; formals = [(String, "x"); (String,"y")];
    locals = []; body = [] } (StringMap.add "find"
  { typ = String; fname = "find"; formals = [(String, "x"); (String, "y")];
    locals = []; body = [] } (StringMap.add "strcmp"
  { typ = Int; fname = "strcmp"; formals = [(String, "x"); (String, "y")];
    locals = []; body = [] } (StringMap.singleton "len"
  { typ = Int; fname = "len"; formals = [(String, "x")];
```

```
    locals = []; body = [] } ))))))))))))))))

in

let function_decls =
       List.fold_left (fun m fd -> StringMap.add fd.fname fd m) built_in_decls functions
in

let function_decl s = try StringMap.find s function_decls
       with Not_found -> raise (Failure ("Unrecognized function " ^ s))
in

let _ = function_decl "main" in

(* A function that is used to check each function *)
let check_function func =


       List.iter(check_not_void (fun n ->
              "Illegal void formal " ^ n ^ " in " ^ func.fname)) func.formals;

       report_duplicate (fun n ->
              "Duplicate formal " ^ n ^ " in " ^ func.fname)(List.map snd func.formals);

       List.iter (check_not_void (fun n ->
              "Illegal void local " ^ n ^ " in " ^ func.fname)) func.locals;

       report_duplicate (fun n ->
              "Duplicate local " ^ n ^ " in " ^ func.fname)(List.map snd func.locals);

(* Check variables *)
  let symbols = List.fold_left (fun m (t, n) -> StringMap.add n t m) (* name type map *)
    StringMap.empty (globals @ func.formals @ func.locals )
  in

  let symbols = ref symbols in


 let type_of_tuple t =
    match (List.hd t) with
      IntLit _ -> TupleTyp(Int, List.length t)
    | _ -> raise (Failure ("illegal tuple type")) in

let find_rowtyp name m =
       let m = StringMap.find m !symbols in
       let typ = match m with
              MatrixTyp(Int, _, _) -> Int
         | MatrixTyp(Float, _, _) -> Float
         | MatrixTyp(TupleTyp(Int, len), _, _) -> TupleTyp(Int, len)
```

```
        | _ -> raise (Failure ("illegal matrix type")) in
      let cols = match m with
            MatrixTyp(_, _, c) -> c
        | _ -> raise (Failure ("illegal matrix type")) in
      symbols := StringMap.add name (RowTyp(typ, cols)) !symbols in

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

  let rec check_tuple_literal tt l i =
    let length = List.length l in
    match (tt, List.nth l i) with
      (TupleTyp(Int, _), IntLit _) -> if i == length - 1 then TupleTyp(Int, length) else
check_tuple_literal (TupleTyp(Int, length)) l (succ i)
    | _ -> raise (Failure ("illegal tuple literal"))
  in

  let tuple_access_type = function
      TupleTyp(p, _) -> p
    | _ -> raise (Failure ("illegal tuple access")) in

  let row_access_type = function
      RowTyp(r, _) -> r
    | _ -> raise (Failure ("illegal row access")) in

  let matrix_access_type = function
      MatrixTyp(t, _, _) -> t
    | _ -> raise (Failure ("illegal matrix access") ) in

  let mrow_access_type = function
          MatrixTyp(t, _, c) -> RowTyp(t, c)
        | _ -> raise (Failure ("illegal matrix access") ) in


  let type_of_row r l =
       match (List.hd r) with
        IntLit _ -> RowTyp(Int, l)
      | FloatLit _ -> RowTyp(Float, l)
      | TupleLit t -> RowTyp((type_of_tuple) t, l)
      | _ -> raise (Failure ("illegal row type"))
  in

  let type_of_matrix m r c =
    match (List.hd (List.hd m)) with
        IntLit _ -> MatrixTyp(Int, r, c)
      | FloatLit _ -> MatrixTyp(Float, r, c)
      | TupleLit t -> MatrixTyp((type_of_tuple) t, r, c)
```

```
      | _ -> raise (Failure ("illegal matrix type"))
  in

 let check_pointer_type = function
      RowPointer(t) -> RowPointer(t)
    | MatrixPointer(t) -> MatrixPointer(t)
    | _ -> raise ( Failure ("cannot increment a non-pointer type") )
  in

    let matrix_type s = match (List.hd s) with
    | IntLit _ -> RowTyp(Int, List.length s)
    | FloatLit _ -> RowTyp(Float, List.length s)
    | BoolLit _ -> RowTyp(Bool, List.length s)
    | _ -> raise ( Failure ("Cannot instantiate a matrix of that type")) in

    let rec check_all_matrix_literal m ty idx =
      let length = List.length m in
        match (ty, List.nth m idx) with
      (RowTyp(Int, _), IntLit _) -> if idx == length - 1 then RowTyp(Int, length) else
check_all_matrix_literal m (RowTyp(Int, length)) (succ idx)
    | (RowTyp(Float, _), FloatLit _) -> if idx == length - 1 then RowTyp(Float, length) else
check_all_matrix_literal m (RowTyp(Float, length)) (succ idx)
    | (RowTyp(Bool, _), BoolLit _) -> if idx == length - 1 then RowTyp(Bool, length) else
check_all_matrix_literal m (RowTyp(Bool, length)) (succ idx)
    | _ -> raise (Failure ("illegal matrix literal"))
  in


  let check_row_pointer_type = function
    RowTyp(p, _) -> RowPointer(p)
    | _ -> raise ( Failure ("cannot reference non-row pointer type"))
  in

  let check_matrix_pointer_type = function
      MatrixTyp(p, _, _) -> MatrixPointer(p)
    | _ -> raise ( Failure ("cannot reference a non-matrix pointer type"))
  in

  let pointer_type = function
    | RowPointer(t) -> t
    | MatrixPointer(t) -> t
    | _ -> raise ( Failure ("cannot dereference a non-pointer type") ) in

  (* Return the type of an expression or throw an exception *)
  let rec expr = function
    IntLit _ -> Int
  | FloatLit _ -> Float
  | StringLit _ -> String
  | BoolLit _ -> Bool
```

```
  | Id s -> type_of_identifier s
  | Null(e) -> let k=expr e in (match k with Void -> raise(Failure("no void expression
expected"))
                              |_-> Bool)
  | RowLit r -> type_of_row r (List.length r)
  | TupleLit t -> check_tuple_literal (type_of_tuple t) t 0
  | MatrixLit m -> type_of_matrix m (List.length m) (List.length (List.hd m))
  | RowAccess(s, e) -> let _ = (match (expr e) with
                                  Int -> Int
                                | _ -> raise (Failure ("attempting to access with
non-integer type"))) in
                           row_access_type (type_of_identifier s)
  | TupleAccess(s, e) -> let _ = (match (expr e) with
                                  Int -> Int
                                | _ -> raise (Failure ("attempting to access with a
non-integer type"))) in
                         tuple_access_type (type_of_identifier s)
  | MatrixAccess(s, e1, e2) -> let _ = (match (expr e1) with
                                      Int -> Int
                                    | _ -> raise (Failure ("attempting to access with a
non-integer type")))
                               and _ = (match (expr e2) with
                                      Int -> Int
                                    | _ -> raise (Failure ("attempting to access with a
non-integer type"))) in
                               matrix_access_type (type_of_identifier s)
  | MRowAccess(s, e) -> let _ = (match (expr e) with
                                                          Int -> Int
                                                        | _ -> raise (Failure
("attempting to access with non-integer type"))) in
                                                        mrow_access_type (type_of_identifier s)
  | Length(s) -> let typ = (match (type_of_identifier s) with
                                                        MatrixTyp(_, _, _) -> Int
                                                    |    RowTyp(_, _) -> Int
                                                    | _ -> raise (Failure ("attempting to get
length of wrong type"))) in typ
  | Width(s) -> let typ = (match (type_of_identifier s) with
                                                        MatrixTyp(_,_,_) -> Int
                                                    | _ -> raise (Failure ("attempting to get
width of wrong type"))) in typ
  | Type(s) -> let typ = (match (type_of_identifier s) with
                                                        MatrixTyp(_, _, _) -> String
                                                  | RowTyp(_, _) -> String
                                                    | _ -> raise (Failure ("attempting to get type
of a non-matrix or row"))) in typ
  | RowReference(s) -> check_row_pointer_type( type_of_identifier s )
  | PointerIncrement(s) -> check_pointer_type (type_of_identifier s)
  | Dereference(s) -> pointer_type (type_of_identifier s)
  | MatrixReference(s) -> check_matrix_pointer_type (type_of_identifier s)
```

```
  | Binop(e1, op, e2) as e -> let t1 = expr e1 and t2 = expr e2 in
  (match op with
    Add -> (match t1,t2 with Int,Int -> Int
      | Float,Float -> Float
      | TupleTyp(Int,l1),TupleTyp(Int,l2) when l1=l2 -> TupleTyp(Int,l1)
      | TupleTyp(Int,l1), Int -> TupleTyp(Int, l1)
      | Int, TupleTyp(Int,l1) -> TupleTyp(Int, l1)
      | MatrixTyp(Int,r1,c1),MatrixTyp(Int,r2,c2) when r1=r2 && c1=c2 ->
MatrixTyp(Int,r1,c1)
      | MatrixTyp(Int,r1,c1), Int -> MatrixTyp(Int,r1,c1)
      | Int, MatrixTyp(Int,r1,c1) -> MatrixTyp(Int,r1,c1)
      | MatrixTyp(Float,r1,c1),MatrixTyp(Float,r2,c2) when r1=r2 && c1=c2 ->
MatrixTyp(Float,r1,c1)
      | MatrixTyp(Float,r1,c1), Float -> MatrixTyp(Float,r1,c1)
      | Float, MatrixTyp(Float,r1,c1) -> MatrixTyp(Float,r1,c1)
      | _,_ -> raise (Failure("illegal addition operator")))
    | Sub -> (match t1,t2 with Int,Int -> Int
      | Float,Float -> Float
      | TupleTyp(Int,l1),TupleTyp(Int,l2) when l1=l2 -> TupleTyp(Int,l1)
      | TupleTyp(Int,l1), Int -> TupleTyp(Int, l1)
      | Int, TupleTyp(Int,l1) -> TupleTyp(Int, l1)
      | MatrixTyp(Int,r1,c1),MatrixTyp(Int,r2,c2) when r1=r2 && c1=c2 ->
MatrixTyp(Int,r1,c1)
      | MatrixTyp(Int,r1,c1), Int -> MatrixTyp(Int,r1,c1)
      | Int, MatrixTyp(Int,r1,c1) -> MatrixTyp(Int,r1,c1)
      | MatrixTyp(Float,r1,c1),MatrixTyp(Float,r2,c2) when r1=r2 && c1=c2 ->
MatrixTyp(Float,r1,c1)
      | MatrixTyp(Float,r1,c1), Float -> MatrixTyp(Float,r1,c1)
      | Float, MatrixTyp(Float,r1,c1) -> MatrixTyp(Float,r1,c1)
      | _,_ -> raise (Failure("illegal subtraction operator")))
    | Mult -> (match t1,t2 with Int,Int -> Int
      | Float,Float -> Float
      | TupleTyp(Int, l1), Int -> TupleTyp(Int, l1)
      | Int, TupleTyp(Int, l1) -> TupleTyp(Int, l1)
      | Int, MatrixTyp(Int,r1,c1) -> MatrixTyp(Int,r1,c1)
      | MatrixTyp(Int,r1,c1), Int -> MatrixTyp(Int,r1,c1)
      | Float, MatrixTyp(Float,r1,c1) -> MatrixTyp(Float,r1,c1)
      | MatrixTyp(Float,r1,c1), Float -> MatrixTyp(Float,r1,c1)
      | _,_ -> raise (Failure("illegal multiplication operator")))
    | Div -> (match t1,t2 with Int,Int -> Int
      | Float,Float -> Float
      | TupleTyp(Int, l1), Int -> TupleTyp(Int, l1)
      | Int, TupleTyp(Int, l1) -> TupleTyp(Int, l1)
      | Int, MatrixTyp(Int,r1,c1) -> MatrixTyp(Int,r1,c1)
      | MatrixTyp(Int,r1,c1), Int -> MatrixTyp(Int,r1,c1)
      | Float, MatrixTyp(Float,r1,c1) -> MatrixTyp(Float,r1,c1)
      | MatrixTyp(Float,r1,c1), Float -> MatrixTyp(Float,r1,c1)
      | _,_ -> raise (Failure("illegal division operator")))
    | Equal | Neq when t1 = t2 -> Bool
```

```
    | Less | Leq | Greater | Geq when t1 = Int && t2 = Int -> Bool
    | Less | Leq | Greater | Geq when t1 = Float && t2 = Float -> Float
    | And | Or when t1 = Bool && t2 = Bool -> Bool
    | _ -> raise (Failure ("illegal binary operator " ^
      string_of_typ t1 ^ " " ^ string_of_op op ^ " " ^
      string_of_typ t2 ^ " in " ^ string_of_expr e))
  )
  | Unop(op, e) as ex -> let t = expr e in
  (match op with
      Neg when t = Int -> Int
    | Neg when t = Float -> Float
    | Not when t = Bool -> Bool
    | _ -> raise (Failure ("illegal unary operator " ^ string_of_uop op ^
      string_of_typ t ^ " in " ^ string_of_expr ex)))
  | Init(var, lit) -> let a = type_of_identifier var and b= expr lit in
    (match b with Int -> a
    | _ -> raise (Failure("illegal "^ string_of_typ b ^", expected int")))
  | Noexpr -> Void
  | Assign(e1, e2) as ex -> let lt = (match e1 with
                                      | RowAccess(s, _) -> (match (type_of_identifier s)
with
                                                              RowTyp(t, _) -> (match
t with

Int -> Int

| Float -> Float

                                                                                    |
TupleTyp(p, l) -> TupleTyp(p, l)

| _ -> raise ( Failure ("illegal row") )

)
                                                            | _ -> raise ( Failure
("cannot access a primitive") )
                                                    )
                                      | TupleAccess(s, e) -> (match (expr e) with
                                                              Int -> (match
(type_of_identifier s) with

TupleTyp(p, _) -> (match p with

Int -> Int


              | _ -> raise ( Failure ("illegal datatype" ))

)
```

```
                                                                                | _ ->
raise ( Failure ("cannot access a non-tuple type") )
                                                                                )
                                                                    | _ -> raise ( Failure
("expression is not of type int") )
                                                                )
                                        | MatrixAccess(s, _, _) -> (match (type_of_identifier
s) with
                                                                    MatrixTyp(t, _, _) ->
(match t with

Int -> Int

| Float -> Float

| TupleTyp(p, l) -> TupleTyp(p, l)

| _ -> raise ( Failure ("illegal matrix of matrices") )

)
                                                                    | _ -> raise ( Failure
("cannot access a primitive") )
                                                            )
                                        | _ -> expr e1)
                                and rt = (match e2 with
                                        | RowAccess(s, _) -> (match (type_of_identifier s)
with
                                                                    RowTyp(t, _) -> (match
t with

Int -> Int

| Float -> Float

| TupleTyp(p, l) -> TupleTyp(p, l)

| _ -> raise ( Failure ("illegal row") )

)
                                                                    | _ -> raise ( Failure
("cannot access a primitive") )
                                                            )
                                        | TupleAccess(s, e) -> (match (expr e) with
                                                                Int -> (match
(type_of_identifier s) with

TupleTyp(p, _) -> (match p with

Int -> Int
```

```
              | _ -> raise ( Failure("illegal datatype"))

)
                                                                                | _ ->
raise ( Failure ("cannot access a non-tuple type") )
                                                                              )
                                                                  | _ -> raise ( Failure
("expression is not of datatype int") )
                                                               )
                                        | MatrixAccess(s, _, _) -> (match (type_of_identifier
s) with
                                                                    MatrixTyp(t, _, _) ->
(match t with


Int -> Int

| Float -> Float

| TupleTyp(p, l) -> TupleTyp(p, l)

| _ -> raise ( Failure ("illegal matrix of matrices") )

)
                                                                  | _ -> raise ( Failure
("cannot access a primitive") )
                                                            )
                                        | _ -> expr e2) in
  check_assign lt rt (Failure ("illegal assignment " ^ string_of_typ lt ^
    " = " ^ string_of_typ rt ^ " in " ^
  string_of_expr ex))
  | Call(fname, actuals) as call -> let fd = function_decl fname in
  if List.length actuals != List.length fd.formals then
   raise (Failure ("expecting " ^ string_of_int
     (List.length fd.formals) ^ " arguments in " ^ string_of_expr call))
  else
   List.iter2 (fun (ft, _) e -> let et = expr e in
    ignore (check_assign ft et
      (Failure ("illegal actual argument found " ^ string_of_typ et ^
        " expected " ^ string_of_typ ft ^ " in " ^ string_of_expr e))))
   fd.formals actuals;
   fd.typ
  | _ -> raise (Failure ("unexpected type of expression"))

  in

  let check_bool_expr e =
```

```
    match (expr e) with
      Bool -> ()
    | _ -> raise (Failure ("expected Boolean expression in " ^ string_of_expr e))
  in

  (* Verify a statement or throw an exception *)
  let rec stmt = function
  Block sl -> let rec check_block = function
  [Return _ as s] -> stmt s
  | Return _ :: _ -> raise (Failure "nothing may follow a return")
  | Block sl :: ss -> check_block (sl @ ss)
  | s :: ss -> stmt s ; check_block ss
  | [] -> ()
  in check_block sl
  | Expr e -> ignore (expr e)
  | Return e -> let t = expr e in if t = func.typ then () else
  raise (Failure ("return gives " ^ string_of_typ t ^ " expected " ^
   string_of_typ func.typ ^ " in " ^ string_of_expr e))

  | If(p, b1, b2) -> check_bool_expr p; stmt b1; stmt b2
  | For(e1, e2, e3, st) -> ignore (expr e1); check_bool_expr e2;
  ignore (expr e3); stmt st
  | MFor(s1, s2, s) -> find_rowtyp s1 s2; ignore(s1); ignore(s2); stmt s
  | While(p, s) -> check_bool_expr p; stmt s
  in

  stmt (Block func.body)

  in
  List.iter check_function functions
```

e.

exceptions.ml

```
exception UnsupportedMatrixType
exception UnsupportedRowType
exception UnsupportedTupleType

exception UnsupportedBinop
exception UnsupportedUnopOnFloat
exception UnsupportedUnop

exception IllegalAssignment
exception UnsupportedReturnType

exception MatrixOutOfBoundsAccess
exception IllegalMatrixBinop
exception IllegalPointerType
```

f.

codegen.ml

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

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

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

Detailed documentation on the OCaml LLVM library:

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

*)

module L = Llvm
module A = Ast
open Exceptions

module StringMap = Map.Make(String)

let translate (globals, functions) =
  let context = L.global_context () in
  let the_module = L.create_module context "JSTEM"
  and i32_t  = L.i32_type  context
  and float_t   = L.double_type context
  and i8_t   = L.i8_type   context
  and pointer_t = L.pointer_type
  and array_t   = L.array_type
  and i1_t   = L.i1_type   context
  and void_t = L.void_type context in

  let ltype_of_typ = function
      A.Int -> i32_t
    | A.Bool -> i1_t
    | A.Void -> void_t
    | A.Float -> float_t
    | A.String -> pointer_t i8_t
    | A.TupleTyp(typ, size) -> (match typ with
                                  A.Int    -> array_t i32_t size
                                | _ -> raise (UnsupportedTupleType))
    | A.MatrixTyp(typ, size1, size2) -> (match typ with
                                      A.Int    -> array_t (array_t i32_t size2) size1
                                    | A.Float  -> array_t (array_t float_t size2) size1
                                    | A.TupleTyp(typ1,size3) -> (match typ1 with
                                             A.Int    -> array_t (array_t (array_t i32_t
```

```
size3) size2) size1
                                          | _ -> raise (UnsupportedTupleType))
                                    | _ -> raise (UnsupportedMatrixType))
    | A.RowTyp(typ, size) -> (match typ with
                                      A.Int    -> array_t i32_t size
                                    | A.Float  -> array_t float_t size
                                    | A.TupleTyp(typ1,size1) -> (match typ1 with
                                             A.Int    -> array_t (array_t i32_t size1) size
                                           | _ -> raise (UnsupportedTupleType))
                                    | _ -> raise (UnsupportedRowType))
    | A.RowPointer(t) -> (match t with
                              A.Int -> pointer_t i32_t
                            | A.Float -> pointer_t float_t
                                                 | A.TupleTyp(typ1,size1) -> (match typ1
with
                                               A.Int    -> pointer_t (array_t i32_t size1)
                                             | _ -> raise (UnsupportedTupleType))
                            | _ -> raise (IllegalPointerType))
    | A.MatrixPointer(t) -> (match t with
                              A.Int -> pointer_t i32_t
                            | A.Float -> pointer_t float_t
                                                 | A.TupleTyp(typ1,size1) -> (match typ1
with
                                               A.Int    -> pointer_t (array_t i32_t size1)
                                             | _ -> raise (UnsupportedTupleType))
                            | _ -> raise (IllegalPointerType))
    | _ -> raise (Failure("TypeNotFound"))
  in


  (* Declare each global variable; remember its value in a map *)
  let global_vars =
    let global_var m (t, n) =
      let init = L.const_int (ltype_of_typ t) 0
      in StringMap.add n (L.define_global n init the_module) m in
    List.fold_left global_var StringMap.empty globals in

  (* Printing *)
  let printf_t = L.var_arg_function_type i32_t [| L.pointer_type i8_t |] in
  let printf_func = L.declare_function "printf" printf_t the_module in

  let prints_t = L.var_arg_function_type (pointer_t i8_t) [|L.pointer_type i8_t|] in
  let prints_func = L.declare_function "puts" prints_t the_module in

  (* file inegration *)
  let open_ty = L.function_type (pointer_t i8_t) [| (pointer_t i8_t) ; L.pointer_type i8_t
|] in
  let open_func = L.declare_function "fopen" open_ty the_module in
  let close_ty = L.function_type i32_t [| (pointer_t i8_t) |] in
```

```
  let close_func = L.declare_function "fclose" close_ty the_module in
  let read_ty = L.function_type i32_t [| (pointer_t i8_t); i32_t; i32_t; (pointer_t i8_t) |]
in
  let read_func = L.declare_function "fread" read_ty the_module in
  let get_ty = L.function_type (pointer_t i8_t) [| (pointer_t i8_t); i32_t; (pointer_t i8_t)
|] in
  let get_func = L.declare_function "fgets" get_ty the_module in
  let fwrite_ty = L.function_type i32_t [| (pointer_t i8_t); i32_t; i32_t; (pointer_t i8_t)
|] in
  let fwrite_func = L.declare_function "fwrite" fwrite_ty the_module in

  (* String functions *)
  let strlen_ty = L.function_type i32_t [| (pointer_t i8_t) |] in
  let strlen_func = L.declare_function "strlen" strlen_ty the_module in
  let cast_str_int_ty = L.function_type i32_t [| (pointer_t i8_t) |] in
  let cast_str_int_func = L.declare_function "atoi" cast_str_int_ty the_module in
  let sprintf_ty = L.function_type (pointer_t i8_t) [| (pointer_t i8_t); L.pointer_type
i8_t; i32_t|] in
  let sprintf_func = L.declare_function "sprintf" sprintf_ty the_module in
  let calloc_ty = L.function_type (pointer_t i8_t) [|i32_t; i32_t|] in
  let calloc_func = L.declare_function "calloc" calloc_ty the_module in
  let strtok_ty = L.function_type (pointer_t i8_t) [| (pointer_t i8_t) ; (pointer_t i8_t) |]
in
  let strtok_fun = L.declare_function "strtok" strtok_ty the_module in
  let strfind_ty = L.function_type (pointer_t i8_t) [| (pointer_t i8_t); (pointer_t i8_t)|]
in
  let strfind_func = L.declare_function "strstr" strfind_ty the_module in
  let strcmp_ty = L.function_type i32_t [| (pointer_t i8_t); (pointer_t i8_t)|] in
  let strcmp_func = L.declare_function "strcmp" strcmp_ty the_module in

  (* Data casting *)

  (* Define each function (arguments and return type) so we can call it *)
  let function_decls =
    let function_decl m fdecl =
      let name = fdecl.A.fname
      and formal_types =
  Array.of_list (List.map (fun (t,_) -> ltype_of_typ t) fdecl.A.formals)
      in let ftype = L.function_type (ltype_of_typ fdecl.A.typ) 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.A.fname 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" "fmt" builder in
    let float_format_str = L.build_global_stringptr "%f" "fmt" builder in
    let char_format_str = L.build_global_stringptr "%c" "fmt" builder in
```

```
    (* Construct the function's "locals": formal arguments and locally
       declared variables.  Allocate each on the stack, initialize their
       value, if appropriate, and remember their values in the "locals" map *)

  let types = ref StringMap.empty in

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

      let add_local m (t, n) =

      types := StringMap.add n t !types;

  let local_var =
   (match t with
       A.MatrixTyp(A.TupleTyp(_, l), r, c) -> if r * c * l > 1000
                                              then L.build_malloc (ltype_of_typ t) n
builder
                                            else
                                              L.build_alloca (ltype_of_typ t) n builder
     | _ -> L.build_alloca (ltype_of_typ t) n builder)
  in StringMap.add n local_var m in

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

       let local_vars = ref local_vars in

       (* ADD INDEX VARIABLE TO LOCALS FOR MATRIX FOR LOOP *)
       local_vars := StringMap.add "_i" (L.build_alloca (ltype_of_typ A.Int) "_i" builder)
!local_vars;

       let check_function = List.fold_left (fun m (t, n) -> StringMap.add n t m)
              StringMap.empty (globals @ fdecl.A.formals @ fdecl.A.locals) in

       let check_function = ref check_function in


       (* ADD ROW TO LOCALS FOR MATRIX FOR LOOP *)
       let allocate_row name len m =

    let typ = match (StringMap.find m !types) with
      A.MatrixTyp(A.Int, _, _) -> A.Int
      | A.MatrixTyp(A.Float, _, _) -> A.Float
```

```
      | A.MatrixTyp(A.TupleTyp(A.Int, len), _, _) -> A.TupleTyp(A.Int, len)
      | _ -> raise (Failure ("illegal matrix type")) in

              check_function := StringMap.add name (A.RowTyp(typ, len)) !check_function;
              check_function := StringMap.add "_i" (A.Int) !check_function;
              local_vars := StringMap.add name (L.build_alloca (ltype_of_typ (A.RowTyp(typ,
len))) name builder) !local_vars in

    (* Return the value for a variable or formal argument *)
    let lookup n =
                                    try StringMap.find n !local_vars
                  with Not_found -> StringMap.find n global_vars
    in

    let type_of_identifier s =
      let symbols = !check_function in
      StringMap.find s symbols
    in

    let build_row_argument s builder =
      L.build_in_bounds_gep (lookup s) [| L.const_int i32_t 0; L.const_int i32_t 0 |] s
builder
    in

    let build_matrix_argument s builder =
      L.build_in_bounds_gep (lookup s) [| L.const_int i32_t 0; L.const_int i32_t 0;
L.const_int i32_t 0 |] s builder
    in

    let build_pointer_dereference s builder isAssign =
      if isAssign
        then L.build_load (lookup s) s builder
      else
        L.build_load (L.build_load (lookup s) s builder) s builder
    in

    let build_pointer_increment s builder isAssign =
      if isAssign
        then L.build_load (L.build_in_bounds_gep (lookup s) [| L.const_int i32_t 1 |] s
builder) s builder
      else
        L.build_in_bounds_gep (L.build_load (L.build_in_bounds_gep (lookup s) [| L.const_int
i32_t 0 |] s builder) s builder) [| L.const_int i32_t 1 |] s builder
    in

    let get_tuple_type tuple =
      match (List.hd tuple) with
        A.IntLit _ -> ltype_of_typ (A.Int)
      | _ -> raise (UnsupportedTupleType) in
```

```
   let get_matrix_type matrix =
     match (List.hd matrix) with
       A.IntLit _ -> ltype_of_typ (A.Int)
     | A.FloatLit _ -> ltype_of_typ (A.Float)
     | A.TupleLit _ -> ltype_of_typ (A.TupleTyp(A.Int, 3))
     | _ -> raise (UnsupportedMatrixType) in

   let get_row_type row =
     match (List.hd row) with
       A.IntLit _ -> ltype_of_typ (A.Int)
     | A.FloatLit _ -> ltype_of_typ (A.Float)
     | A.TupleLit _ -> ltype_of_typ (A.TupleTyp(A.Int, 3))
     | _ -> raise (UnsupportedRowType) in

 let build_row_access s i1 i2 builder isAssign =
   if isAssign
   then L.build_gep (lookup s) [| i1; i2 |] s builder
   else
   L.build_load (L.build_gep (lookup s) [| i1; i2 |] s builder) s builder
 in

 let build_tuple_access s i1 i2 builder isAssign =
   if isAssign
   then L.build_gep (lookup s) [| i1; i2 |] s builder
   else
   L.build_load (L.build_gep (lookup s) [| i1; i2 |] s builder) s builder
 in

 let build_matrix_access s i1 i2 i3 builder isAssign =
   if isAssign
   then L.build_gep (lookup s) [| i1; i2; i3|] s builder
   else
   L.build_load (L.build_gep (lookup s) [| i1; i2; i3 |] s builder) s builder
 in

   (* New code/supporting functions for getting type *)
   (* A function that is used to check each function *)
      let build_mrow_access s i1 i2 builder isAssign =
        if isAssign
             then L.build_gep (lookup s) [| i1; i2 |] s builder
        else
             L.build_load (L.build_gep (lookup s) [| i1; i2 |] s builder) s builder
      in

let type_of_tuple1 t =
   match (List.hd t) with
     A.IntLit _ -> A.TupleTyp(A.Int, List.length t)
   | _ -> raise (Failure ("illegal tuple type")) in
```

```
  let rec check_tuple_literal1 tt l i =
    let length = List.length l in
    match (tt, List.nth l i) with
      (A.TupleTyp(A.Int, _), A.IntLit _) -> if i == length - 1 then A.TupleTyp(A.Int,
length) else check_tuple_literal1 (A.TupleTyp(A.Int, length)) l (succ i)
    | _ -> raise (Failure ("illegal tuple literal"))
  in

  let type_of_row1 r l =
  match (List.hd r) with
        A.IntLit _ -> A.RowTyp(A.Int, l)
      | A.FloatLit _ -> A.RowTyp(A.Float, l)
      | A.TupleLit t -> A.RowTyp((type_of_tuple1) t, l)
      | _ -> raise (Failure ("illegal row type"))
  in

  let type_of_matrix1 m r c =
    match (List.hd (List.hd m)) with
        A.IntLit _ -> A.MatrixTyp(A.Int, r, c)
      | A.FloatLit _ -> A.MatrixTyp(A.Float, r, c)
      | A.TupleLit t -> A.MatrixTyp((type_of_tuple1) t, r, c)
      | _ -> raise (Failure ("illegal matrix type"))
  in

  let rec expr1 = function
    A.IntLit _ -> A.Int
  | A.FloatLit _ -> A.Float
  | A.StringLit _ -> A.String
  | A.BoolLit _ -> A.Bool
  | A.Id s -> type_of_identifier s
  | A.RowLit r -> type_of_row1 r (List.length r)
  | A.TupleLit t -> check_tuple_literal1 (type_of_tuple1 t) t 0
  | A.MatrixLit m -> type_of_matrix1 m (List.length m) (List.length (List.hd m))
  | A.Binop(e1, op, e2) -> let t1 = expr1 e1 and t2 = expr1 e2 in
  (match op with
        A.Add -> (match t1,t2 with A.Int,A.Int -> A.Int
      | A.Float,A.Float | A.Int,A.Float | A.Float,A.Int -> A.Float
      | A.TupleTyp(A.Int,l1),A.TupleTyp(A.Int,l2) when l1=l2 -> A.TupleTyp(A.Int,l1)
      | A.MatrixTyp(A.Int,r1,c1),A.MatrixTyp(A.Int,r2,c2) when r1=r2 && c1=c2 ->
A.MatrixTyp(A.Int,r1,c1)
      | A.MatrixTyp(A.Float,r1,c1),A.MatrixTyp(A.Float,r2,c2) when r1=r2 && c1=c2 ->
A.MatrixTyp(A.Float,r1,c1)
      | _,_ -> raise (Failure("illegal type")))
    | A.Sub -> (match t1,t2 with A.Int,A.Int -> A.Int
      | A.Float,A.Float | A.Int,A.Float | A.Float,A.Int -> A.Float
      | A.TupleTyp(A.Int,l1),A.TupleTyp(A.Int,l2) when l1=l2 -> A.TupleTyp(A.Int,l1)
      | A.MatrixTyp(A.Int,r1,c1),A.MatrixTyp(A.Int,r2,c2) when r1=r2 && c1=c2 ->
A.MatrixTyp(A.Int,r1,c1)
```

```
      | A.MatrixTyp(A.Float,r1,c1),A.MatrixTyp(A.Float,r2,c2) when r1=r2 && c1=c2 ->
A.MatrixTyp(A.Float,r1,c1)
      | _,_ -> raise (Failure("illegal type")))
    | A.Mult -> (match t1,t2 with A.Int,A.Int -> A.Int
      | A.Float,A.Float | A.Int,A.Float | A.Float,A.Int -> A.Float
      | _,_ -> raise (Failure("illegal type")))
    | A.Div -> (match t1,t2 with A.Int,A.Int -> A.Int
      | A.Float,A.Float | A.Int,A.Float | A.Float,A.Int -> A.Float
      | _,_ -> raise (Failure("illegal type")))
    | _ -> raise (Failure ("illegal binop")))
  | A.Unop(op, e) -> let t = expr1 e in
  (match op with
      A.Neg when t = A.Int -> A.Int
    | A.Neg when t = A.Float -> A.Float
    | A.Not when t = A.Bool -> A.Bool
    | _ -> raise (Failure ("illegal unop")))
  | A.Call(_,_) -> A.Int
  | _ -> raise (Failure("illegal expression"))
  in

  let build_mrow_access s i1 i2 builder isAssign =
    if isAssign
    then L.build_gep (lookup s) [| i1; i2 |] s builder
    else
    L.build_load (L.build_gep (lookup s) [| i1; i2 |] s builder) s builder
  in

  let build_matrix_access s i1 i2 i3 builder isAssign =
    if isAssign
    then L.build_gep (lookup s) [| i1; i2; i3|] s builder
    else
    L.build_load (L.build_gep (lookup s) [| i1; i2; i3 |] s builder) s builder
  in

  let build_mrow_access s i1 i2 builder isAssign =
    if isAssign
    then L.build_gep (lookup s) [| i1; i2 |] s builder
    else
    L.build_load (L.build_gep (lookup s) [| i1; i2 |] s builder) s builder
  in

       let get_length s =
              L.array_length (L.type_of (L.build_load (L.build_gep (lookup s) [|
L.const_int i32_t 0 |] s builder) s builder)) in

       let get_width s =
              L.array_length (L.type_of (L.build_load (L.build_gep (lookup s) [|
L.const_int i32_t 0; L.const_int i32_t 0 |] s builder) s builder)) in
```

```
       let get_type s =
              let temp = (match (type_of_identifier s) with
                     A.MatrixTyp(A.Int, _, _) -> "int"
                | A.MatrixTyp(A.Float, _, _) -> "float"
                | A.MatrixTyp(A.TupleTyp(_, _), _, _) -> "tuple"
                | A.RowTyp(A.Int, _) -> "int"
                | A.RowTyp(A.Float, _) -> "float"
                | A.RowTyp(A.TupleTyp(_, _), _) -> "tuple"
                | _ -> raise (Failure("illegal type"))) in temp in

    (* Construct code for an expression; return its value *)
    let rec expr builder = function
        A.IntLit i -> L.const_int i32_t i
      | A.FloatLit f -> L.const_float float_t f
      | A.BoolLit b -> L.const_int i1_t (if b then 1 else 0)
      | A.StringLit s -> L.build_global_stringptr s "string" builder
      | A.Noexpr -> L.const_int i32_t 0
      | A.Id s -> L.build_load (lookup s) s builder
      | A.TupleLit t -> L.const_array (get_tuple_type t) (Array.of_list (List.map (expr
builder) t))
      | A.MatrixLit m -> (match (List.hd (List.hd m)) with
                                A.FloatLit _ -> let realOrder=List.map List.rev m in let
i32Lists = List.map (List.map (expr builder)) realOrder in let listOfArrays=List.map
Array.of_list i32Lists in let i32ListOfArrays = List.map (L.const_array float_t)
listOfArrays in let arrayOfArrays=Array.of_list i32ListOfArrays in L.const_array (array_t
float_t (List.length (List.hd m))) arrayOfArrays
                              | A.IntLit _ -> let realOrder=List.map List.rev m in let
i32Lists = List.map (List.map (expr builder)) realOrder in let listOfArrays=List.map
Array.of_list i32Lists in let i32ListOfArrays = List.map (L.const_array i32_t) listOfArrays
in let arrayOfArrays=Array.of_list i32ListOfArrays in L.const_array (array_t i32_t
(List.length (List.hd m))) arrayOfArrays
                              | A.TupleLit t -> let realOrder=List.map List.rev m in let
i32Lists = List.map (List.map (expr builder)) realOrder in let listOfArrays=List.map
Array.of_list i32Lists in let i32ListOfArrays = List.map (L.const_array (array_t
(get_tuple_type t) (List.length t))) listOfArrays in let arrayOfArrays=Array.of_list
i32ListOfArrays in L.const_array (array_t (array_t (get_tuple_type t) (List.length t))
(List.length (List.hd m))) arrayOfArrays
                              | _ -> raise ( UnsupportedMatrixType ))
      | A.RowReference (s) -> build_row_argument s builder
      | A.MatrixReference (s) -> build_matrix_argument s builder
      | A.RowLit r ->  L.const_array (get_row_type r) (Array.of_list (List.map (expr
builder) r))
           | A.MatrixAccess(s, e1, e2) -> let i1 = expr builder e1 and i2 = expr builder e2
in build_matrix_access s (L.const_int i32_t 0) i1 i2 builder false
      | A.PointerIncrement (s) -> build_pointer_increment s builder false
      | A.Dereference (s) -> build_pointer_dereference s builder false
           | A.RowAccess(s, e1) -> let i1 = expr builder e1 in build_row_access s
(L.const_int i32_t 0) i1 builder false
      | A.TupleAccess(s, e1) -> let i1 = expr builder e1 in build_tuple_access s
```

```
(L.const_int i32_t 0) i1 builder false
      | A.MRowAccess(s, e1) -> let i1 = expr builder e1 in build_mrow_access s (L.const_int
i32_t 0) i1 builder false
      | A.MRowAccess(s, e1) -> let i1 = expr builder e1 in build_mrow_access s (L.const_int
i32_t 0) i1 builder false
           | A.Length(s) -> L.const_int i32_t (get_length s)
                | A.Width(s) -> L.const_int i32_t (get_width s)
                | A.Type(s) -> L.build_global_stringptr (get_type s) "string" builder
      | A.Null(e1) -> let cnt1= expr builder e1 in
        L.build_is_null cnt1 "isnull" builder
      | A.Init(e1,e2) -> let cnt1=(lookup e1) and cnt2= expr builder e2 in
        let tp= L.element_type (L.type_of cnt1) in
        let sz=L.size_of tp in
        let sz1=L.build_intcast sz (i32_t) "intc" builder in
        let dt=L.build_bitcast (L.build_call calloc_func [|cnt2;sz1|] "tmpa" builder) tp
"tmpb" builder in
        L.build_store dt cnt1 builder
      | A.Binop (e1, op, e2) ->
        let e1' = expr builder e1 and
        e2' = expr builder e2 and
        t1 = expr1 e1 and
        t2 = expr1 e2 in
          let float_bop operator =
            (match operator with
              A.Add     -> L.build_fadd
            | A.Sub     -> L.build_fsub
            | A.Mult    -> L.build_fmul
            | A.Div     -> L.build_fdiv
            | A.And     -> L.build_and
            | A.Or      -> L.build_or
            | A.Equal   -> L.build_fcmp L.Fcmp.Oeq
            | A.Neq     -> L.build_fcmp L.Fcmp.One
            | A.Less    -> L.build_fcmp L.Fcmp.Olt
            | A.Leq     -> L.build_fcmp L.Fcmp.Ole
            | A.Greater -> L.build_fcmp L.Fcmp.Ogt
            | A.Geq     -> L.build_fcmp L.Fcmp.Oge
            | _ -> raise (Failure("Unsupported operator"))
            ) e1' e2' "tmp" builder
          in

          let int_bop operator =
            (match operator with
              A.Add     -> L.build_add
            | A.Sub     -> L.build_sub
            | A.Mult    -> L.build_mul
            | A.Div     -> L.build_sdiv
            | A.And     -> L.build_and
            | A.Or      -> L.build_or
            | A.Equal   -> L.build_icmp L.Icmp.Eq
```

```
            | A.Neq     -> L.build_icmp L.Icmp.Ne
            | A.Less    -> L.build_icmp L.Icmp.Slt
            | A.Leq     -> L.build_icmp L.Icmp.Sle
            | A.Greater -> L.build_icmp L.Icmp.Sgt
            | A.Geq     -> L.build_icmp L.Icmp.Sge
            | _ -> raise (Failure("Unsupported operator"))
            ) e1' e2' "tmp" builder
          in

          let tuple_int_bop n_i operator =
            let lhs_str = (match e1 with A.Id(s) -> s | _ -> "") in
            let rhs_str = (match e2 with A.Id(s) -> s | _ -> "") in
              (match operator with
                A.Add ->
                  let tmp_t = L.build_alloca (array_t i32_t n_i) "tmptup" builder in
                  for i=0 to n_i do
                    let v1 = build_tuple_access lhs_str (L.const_int i32_t 0) (L.const_int
i32_t i) builder false in
                    let v2 = build_tuple_access rhs_str (L.const_int i32_t 0) (L.const_int
i32_t i) builder false in
                    let add_res = L.build_add v1 v2 "tmp" builder in
                    let ld = L.build_gep tmp_t [| L.const_int i32_t 0; L.const_int i32_t i
|] "tmptup" builder in
                  ignore(L.build_store add_res ld builder);
                  done;
                L.build_load (L.build_gep tmp_t [| L.const_int i32_t 0 |] "tmptup" builder)
"tmptup" builder
              | A.Sub ->
                  let tmp_t = L.build_alloca (array_t i32_t n_i) "tmptup" builder in
                  for i=0 to n_i do
                    let v1 = build_tuple_access lhs_str (L.const_int i32_t 0) (L.const_int
i32_t i) builder false in
                    let v2 = build_tuple_access rhs_str (L.const_int i32_t 0) (L.const_int
i32_t i) builder false in
                    let add_res = L.build_sub v1 v2 "tmp" builder in
                    let ld = L.build_gep tmp_t [| L.const_int i32_t 0; L.const_int i32_t i
|] "tmptup" builder in
                  ignore(L.build_store add_res ld builder);
                  done;
                L.build_load (L.build_gep tmp_t [| L.const_int i32_t 0 |] "tmptup"
builder) "tmptup" builder
              | _ -> raise (Failure("Unsupported operator")))
            in

          let tuple_int_scalar_bop n_i operator =
            let lhs_str = (match e1 with A.Id(s) -> s | _ -> "") in
            (* let rhs_str = (match e2 with A.Id(s) -> s | _ -> "") in *)
              (match operator with
                A.Add ->
```

```
                  let tmp_t = L.build_alloca (array_t i32_t n_i) "tmptup" builder in
                  for i=0 to n_i do
                    let v1 = build_tuple_access lhs_str (L.const_int i32_t 0) (L.const_int
i32_t i) builder false in
                    let add_res = L.build_add v1 e2' "tmp" builder in
                    let ld = L.build_gep tmp_t [| L.const_int i32_t 0; L.const_int i32_t i
|] "tmptup" builder in
                  ignore(L.build_store add_res ld builder);
                  done;
                L.build_load (L.build_gep tmp_t [| L.const_int i32_t 0 |] "tmptup" builder)
"tmptup" builder
              | A.Sub ->
                  let tmp_t = L.build_alloca (array_t i32_t n_i) "tmptup" builder in
                  for i=0 to n_i do
                    let v1 = build_tuple_access lhs_str (L.const_int i32_t 0) (L.const_int
i32_t i) builder false in
                    let add_res = L.build_sub v1 e2' "tmp" builder in
                    let ld = L.build_gep tmp_t [| L.const_int i32_t 0; L.const_int i32_t i
|] "tmptup" builder in
                  ignore(L.build_store add_res ld builder);
                  done;
                L.build_load (L.build_gep tmp_t [| L.const_int i32_t 0 |] "tmptup"
builder) "tmptup" builder
              | A.Mult ->
                  let tmp_t = L.build_alloca (array_t i32_t n_i) "tmptup" builder in
                  for i=0 to n_i do
                    let v1 = build_tuple_access lhs_str (L.const_int i32_t 0) (L.const_int
i32_t i) builder false in
                    let add_res = L.build_mul v1 e2' "tmp" builder in
                    let ld = L.build_gep tmp_t [| L.const_int i32_t 0; L.const_int i32_t i
|] "tmptup" builder in
                  ignore(L.build_store add_res ld builder);
                  done;
                L.build_load (L.build_gep tmp_t [| L.const_int i32_t 0 |] "tmptup"
builder) "tmptup" builder
              | A.Div ->
                  let tmp_t = L.build_alloca (array_t i32_t n_i) "tmptup" builder in
                  for i=0 to n_i do
                    let v1 = build_tuple_access lhs_str (L.const_int i32_t 0) (L.const_int
i32_t i) builder false in
                    let add_res = L.build_sdiv v1 e2' "tmp" builder in
                    let ld = L.build_gep tmp_t [| L.const_int i32_t 0; L.const_int i32_t i
|] "tmptup" builder in
                  ignore(L.build_store add_res ld builder);
                  done;
                L.build_load (L.build_gep tmp_t [| L.const_int i32_t 0 |] "tmptup"
builder) "tmptup" builder
              | _ -> raise (Failure("Unsupported operator")))
          in
```

```
          let int_tuple_scalar_bop n_i operator =
            let lhs_str = (match e2 with A.Id(s) -> s | _ -> "") in
              (match operator with
                A.Add ->
                  let tmp_t = L.build_alloca (array_t i32_t n_i) "tmptup" builder in
                  for i=0 to n_i do
                    let v1 = build_tuple_access lhs_str (L.const_int i32_t 0) (L.const_int
i32_t i) builder false in
                    let add_res = L.build_add e1' v1 "tmp" builder in
                    let ld = L.build_gep tmp_t [| L.const_int i32_t 0; L.const_int i32_t i
|] "tmptup" builder in
                  ignore(L.build_store add_res ld builder);
                  done;
                L.build_load (L.build_gep tmp_t [| L.const_int i32_t 0 |] "tmptup" builder)
"tmptup" builder
              | A.Sub ->
                  let tmp_t = L.build_alloca (array_t i32_t n_i) "tmptup" builder in
                  for i=0 to n_i do
                    let v1 = build_tuple_access lhs_str (L.const_int i32_t 0) (L.const_int
i32_t i) builder false in
                    let add_res = L.build_sub e1' v1 "tmp" builder in
                    let ld = L.build_gep tmp_t [| L.const_int i32_t 0; L.const_int i32_t i
|] "tmptup" builder in
                  ignore(L.build_store add_res ld builder);
                  done;
                L.build_load (L.build_gep tmp_t [| L.const_int i32_t 0 |] "tmptup"
builder) "tmptup" builder
              | A.Mult ->
                  let tmp_t = L.build_alloca (array_t i32_t n_i) "tmptup" builder in
                  for i=0 to n_i do
                    let v1 = build_tuple_access lhs_str (L.const_int i32_t 0) (L.const_int
i32_t i) builder false in
                    let add_res = L.build_mul e1' v1 "tmp" builder in
                    let ld = L.build_gep tmp_t [| L.const_int i32_t 0; L.const_int i32_t i
|] "tmptup" builder in
                  ignore(L.build_store add_res ld builder);
                  done;
                L.build_load (L.build_gep tmp_t [| L.const_int i32_t 0 |] "tmptup"
builder) "tmptup" builder
              | A.Div ->
                  let tmp_t = L.build_alloca (array_t i32_t n_i) "tmptup" builder in
                  for i=0 to n_i do
                    let v1 = build_tuple_access lhs_str (L.const_int i32_t 0) (L.const_int
i32_t i) builder false in
                    let add_res = L.build_sdiv e1' v1 "tmp" builder in
                    let ld = L.build_gep tmp_t [| L.const_int i32_t 0; L.const_int i32_t i
|] "tmptup" builder in
                  ignore(L.build_store add_res ld builder);
```

```
                    done;
                  L.build_load (L.build_gep tmp_t [| L.const_int i32_t 0 |] "tmptup"
builder) "tmptup" builder
                | _ -> raise (Failure("Unsupported operator")))
              in

            let matrix_int_bop r_i c_i operator =
              let lhs_str = (match e1 with A.Id(s) -> s | _ -> "") in
              let rhs_str = (match e2 with A.Id(s) -> s | _ -> "") in
                (match operator with
                  A.Add ->
                    let tmp_m = L.build_alloca (array_t (array_t i32_t c_i) r_i) "tmpmat"
builder in
                    for i=0 to (r_i-1) do
                      for j=0 to (c_i-1) do
                        let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                        let m2 = build_matrix_access rhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                        let add_res = L.build_add m1 m2 "tmp" builder in
                        let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                      ignore(L.build_store add_res ld builder);
                      done
                    done;
                  L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
                | A.Sub ->
                    let tmp_m = L.build_alloca (array_t (array_t i32_t c_i) r_i) "tmpmat"
builder in
                    for i=0 to (r_i-1) do
                      for j=0 to (c_i-1) do
                        let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                        let m2 = build_matrix_access rhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                        let add_res = L.build_sub m1 m2 "tmp" builder in
                        let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                      ignore(L.build_store add_res ld builder);
                      done
                    done;
                  L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
                | _ -> raise (Failure("Unsupported operator")))
              in

            let matrix_int_scalar_bop r_i c_i operator =
              let lhs_str = (match e1 with A.Id(s) -> s | _ -> "") in
```

```
              (match operator with
                A.Add ->
                  let tmp_m = L.build_alloca (array_t (array_t i32_t c_i) r_i) "tmpmat"
builder in
                  for i=0 to (r_i-1) do
                    for j=0 to (c_i-1) do
                      let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                      let add_res = L.build_add m1 e2' "tmp" builder in
                      let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                    ignore(L.build_store add_res ld builder);
                    done
                  done;
                L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
              | A.Sub ->
                  let tmp_m = L.build_alloca (array_t (array_t i32_t c_i) r_i) "tmpmat"
builder in
                  for i=0 to (r_i-1) do
                    for j=0 to (c_i-1) do
                      let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                      let add_res = L.build_sub m1 e2' "tmp" builder in
                      let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                    ignore(L.build_store add_res ld builder);
                    done
                  done;
                L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
              | A.Mult ->
                  let tmp_m = L.build_alloca (array_t (array_t i32_t c_i) r_i) "tmpmat"
builder in
                  for i=0 to (r_i-1) do
                    for j=0 to (c_i-1) do
                      let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                      let add_res = L.build_mul m1 e2' "tmp" builder in
                      let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                    ignore(L.build_store add_res ld builder);
                    done
                  done;
                L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
              | A.Div ->
                  let tmp_m = L.build_alloca (array_t (array_t i32_t c_i) r_i) "tmpmat"
builder in
```

```
                  for i=0 to (r_i-1) do
                    for j=0 to (c_i-1) do
                      let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                      let add_res = L.build_sdiv m1 e2' "tmp" builder in
                      let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                    ignore(L.build_store add_res ld builder);
                    done
                  done;
                L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
              | _ -> raise (Failure("Unsupported operator")))
           in

           let int_matrix_scalar_bop r_i c_i operator =
             let lhs_str = (match e2 with A.Id(s) -> s | _ -> "") in
               (match operator with
                 A.Add ->
                   let tmp_m = L.build_alloca (array_t (array_t i32_t c_i) r_i) "tmpmat"
builder in
                   for i=0 to (r_i-1) do
                     for j=0 to (c_i-1) do
                       let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                       let add_res = L.build_add e1' m1 "tmp" builder in
                       let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                     ignore(L.build_store add_res ld builder);
                     done
                   done;
                 L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
               | A.Sub ->
                   let tmp_m = L.build_alloca (array_t (array_t i32_t c_i) r_i) "tmpmat"
builder in
                   for i=0 to (r_i-1) do
                     for j=0 to (c_i-1) do
                       let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                       let add_res = L.build_sub e1' m1 "tmp" builder in
                       let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                     ignore(L.build_store add_res ld builder);
                     done
                   done;
                 L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
               | A.Mult ->
```

```
                  let tmp_m = L.build_alloca (array_t (array_t i32_t c_i) r_i) "tmpmat"
builder in
                  for i=0 to (r_i-1) do
                    for j=0 to (c_i-1) do
                      let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                      let add_res = L.build_mul e1' m1 "tmp" builder in
                      let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                    ignore(L.build_store add_res ld builder);
                    done
                  done;
                L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
              | A.Div ->
                  let tmp_m = L.build_alloca (array_t (array_t i32_t c_i) r_i) "tmpmat"
builder in
                  for i=0 to (r_i-1) do
                    for j=0 to (c_i-1) do
                      let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                      let add_res = L.build_sdiv e1' m1 "tmp" builder in
                      let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                    ignore(L.build_store add_res ld builder);
                    done
                  done;
                L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
              | _ -> raise (Failure("Unsupported operator")))
           in

           let matrix_float_bop r_i c_i operator =
             let lhs_str = (match e1 with A.Id(s) -> s | _ -> "") in
             let rhs_str = (match e2 with A.Id(s) -> s | _ -> "") in
               (match operator with
                 A.Add ->
                   let tmp_m = L.build_alloca (array_t (array_t float_t c_i) r_i) "tmpmat"
builder in
                   for i=0 to (r_i-1) do
                     for j=0 to (c_i-1) do
                       let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                       let m2 = build_matrix_access rhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                       let add_res = L.build_fadd m1 m2 "tmp" builder in
                       let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                     ignore(L.build_store add_res ld builder);
```

```
                    done
                  done;
                L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
              | A.Sub ->
                  let tmp_m = L.build_alloca (array_t (array_t float_t c_i) r_i) "tmpmat"
builder in
                  for i=0 to (r_i-1) do
                    for j=0 to (c_i-1) do
                      let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                      let m2 = build_matrix_access rhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                      let add_res = L.build_fsub m1 m2 "tmp" builder in
                      let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                    ignore(L.build_store add_res ld builder);
                    done
                  done;
                L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
              | _ -> raise (Failure("Unsupported operator")))
          in

        let matrix_float_scalar_bop r_i c_i operator =
            let lhs_str = (match e1 with A.Id(s) -> s | _ -> "") in
              (match operator with
                A.Add ->
                  let tmp_m = L.build_alloca (array_t (array_t float_t c_i) r_i) "tmpmat"
builder in
                  for i=0 to (r_i-1) do
                    for j=0 to (c_i-1) do
                      let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                      let add_res = L.build_fadd m1 e2' "tmp" builder in
                      let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                    ignore(L.build_store add_res ld builder);
                    done
                  done;
                L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
              | A.Sub ->
                  let tmp_m = L.build_alloca (array_t (array_t float_t c_i) r_i) "tmpmat"
builder in
                  for i=0 to (r_i-1) do
                    for j=0 to (c_i-1) do
                      let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
```

```
                      let add_res = L.build_fsub m1 e2' "tmp" builder in
                      let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                    ignore(L.build_store add_res ld builder);
                    done
                  done;
                L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
              | A.Mult ->
                  let tmp_m = L.build_alloca (array_t (array_t float_t c_i) r_i) "tmpmat"
builder in
                  for i=0 to (r_i-1) do
                    for j=0 to (c_i-1) do
                      let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                      let add_res = L.build_fmul m1 e2' "tmp" builder in
                      let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                    ignore(L.build_store add_res ld builder);
                    done
                  done;
                L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
              | A.Div ->
                  let tmp_m = L.build_alloca (array_t (array_t float_t c_i) r_i) "tmpmat"
builder in
                  for i=0 to (r_i-1) do
                    for j=0 to (c_i-1) do
                      let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                      let add_res = L.build_fdiv m1 e2' "tmp" builder in
                      let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                    ignore(L.build_store add_res ld builder);
                    done
                  done;
                L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
              | _ -> raise (Failure("Unsupported operator")))
          in

        let float_matrix_scalar_bop r_i c_i operator =
            let lhs_str = (match e2 with A.Id(s) -> s | _ -> "") in
              (match operator with
                A.Add ->
                  let tmp_m = L.build_alloca (array_t (array_t float_t c_i) r_i) "tmpmat"
builder in
                  for i=0 to (r_i-1) do
                    for j=0 to (c_i-1) do
```

```
                      let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                      let add_res = L.build_fadd e1' m1 "tmp" builder in
                      let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                    ignore(L.build_store add_res ld builder);
                    done
                  done;
                L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
              | A.Sub ->
                  let tmp_m = L.build_alloca (array_t (array_t float_t c_i) r_i) "tmpmat"
builder in
                  for i=0 to (r_i-1) do
                    for j=0 to (c_i-1) do
                      let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                      let add_res = L.build_fsub e1' m1 "tmp" builder in
                      let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                    ignore(L.build_store add_res ld builder);
                    done
                  done;
                L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
              | A.Mult ->
                  let tmp_m = L.build_alloca (array_t (array_t float_t c_i) r_i) "tmpmat"
builder in
                  for i=0 to (r_i-1) do
                    for j=0 to (c_i-1) do
                      let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                      let add_res = L.build_fmul e1' m1 "tmp" builder in
                      let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
i; L.const_int i32_t j |] "tmpmat" builder in
                    ignore(L.build_store add_res ld builder);
                    done
                  done;
                L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
              | A.Div ->
                  let tmp_m = L.build_alloca (array_t (array_t float_t c_i) r_i) "tmpmat"
builder in
                  for i=0 to (r_i-1) do
                    for j=0 to (c_i-1) do
                      let m1 = build_matrix_access lhs_str (L.const_int i32_t 0)
(L.const_int i32_t i) (L.const_int i32_t j) builder false in
                      let add_res = L.build_fdiv e1' m1 "tmp" builder in
                      let ld = L.build_gep tmp_m [| L.const_int i32_t 0; L.const_int i32_t
```

```
i; L.const_int i32_t j |] "tmpmat" builder in
                    ignore(L.build_store add_res ld builder);
                    done
                  done;
                L.build_load (L.build_gep tmp_m [| L.const_int i32_t 0 |] "tmpmat" builder)
"tmpmat" builder
              | _ -> raise (Failure("Unsupported operator")))
          in

        let string_of_e1'_llvalue = L.string_of_llvalue e1'
        and string_of_e2'_llvalue = L.string_of_llvalue e2' in

        let space = Str.regexp " " in

        let list_of_e1'_llvalue = Str.split space string_of_e1'_llvalue
        and list_of_e2'_llvalue = Str.split space string_of_e2'_llvalue in

        let i32_re = Str.regexp "i32\\|i32*\\|i8\\|i8*\\|i1\\|i1*"
        and float_re = Str.regexp "double\\|double*" in

        let rec match_string regexp str_list i =
         let length = List.length str_list in
         match (Str.string_match regexp (List.nth str_list i) 0) with
           true -> true
         | false -> if (i > length - 2) then false else match_string regexp str_list (succ
i) in

        let get_type llvalue =
           match (match_string i32_re llvalue 0) with
             true  -> "int"
           | false -> (match (match_string float_re llvalue 0) with
                         true -> "float"
                       | false -> "") in

        let e1'_type = get_type list_of_e1'_llvalue
        and e2'_type = get_type list_of_e2'_llvalue in

        let build_ops_with_types typ1 typ2 =
          match (typ1, typ2) with
            "int", "int" -> (match (t1, t2) with A.Int, A.Int -> int_bop op
                                        | A.TupleTyp(A.Int,l1),A.TupleTyp(A.Int,l2) when
l1=l2->tuple_int_bop l1 op
                                        | A.TupleTyp(A.Int,l1), A.Int -> tuple_int_scalar_bop
l1 op
                                        | A.Int, A.TupleTyp(A.Int,l1) -> int_tuple_scalar_bop
l1 op
                                        | A.MatrixTyp(A.Int,r1,c1),A.MatrixTyp(A.Int,r2,c2)
when r1=r2 && c1=c2 -> matrix_int_bop r1 c1 op
                                        | A.MatrixTyp(A.Int,r1,c1), A.Int ->
```

```
matrix_int_scalar_bop r1 c1 op
                                      | A.Int, A.MatrixTyp(A.Int,r1,c1) ->
int_matrix_scalar_bop r1 c1 op
                                      | A.Bool, A.Bool -> int_bop op
                                      | _,_ -> raise (Failure("Cannot build ops with given
types")))
          | "float" , "float" -> (match (t1,t2) with A.Float, A.Float -> float_bop op
                                      |
A.MatrixTyp(A.Float,r1,c1),A.MatrixTyp(A.Float,r2,c2) when r1=r2 && c1=c2 ->
matrix_float_bop r1 c1 op
                                      | A.MatrixTyp(A.Float,r1,c1), A.Float ->
matrix_float_scalar_bop r1 c1 op
                                      | A.Float, A.MatrixTyp(A.Float,r1,c1) ->
float_matrix_scalar_bop r1 c1 op
                                      | _,_ -> raise (Failure("Cannot build ops with given
types")))
          | _,_ -> raise(UnsupportedBinop)
        in
        build_ops_with_types e1'_type e2'_type
      | A.Unop(op, e) ->
        let e' = expr builder e in

        let float_uops operator =
          match operator with
            A.Neg -> L.build_fneg e' "tmp" builder
          | A.Not -> raise(UnsupportedUnop)  in

        let int_uops operator =
          match operator with
            A.Neg -> L.build_neg e' "tmp" builder
          | A.Not -> L.build_not e' "tmp" builder in

        let bool_uops operator =
          match operator with
          A.Neg -> L.build_neg e' "tmp" builder
          | A.Not -> L.build_not e' "tmp" builder in

        let string_of_e'_llvalue = L.string_of_llvalue e' in

        let space = Str.regexp " " in

        let list_of_e'_llvalue = Str.split space string_of_e'_llvalue in

        let i32_re = Str.regexp "i32\\|i32*"
        and float_re = Str.regexp "double\\|double*"
        and bool_re = Str.regexp "i1\\|i1*" in

        let rec match_string regexp str_list i =
          let length = List.length str_list in
```

```
            match (Str.string_match regexp (List.nth str_list i) 0) with
              true -> true
            | false -> if (i > length - 2) then false else match_string regexp str_list
(succ i) in

        let get_type llvalue =
          match (match_string i32_re llvalue 0) with
            true  -> "int"
          | false -> (match (match_string float_re llvalue 0) with
                 true -> "float"
               | false -> (match (match_string bool_re llvalue 0) with
                             true -> "bool"
                           | false -> "")) in

        let e'_type = get_type list_of_e'_llvalue  in

        let build_ops_with_type typ =
          match typ with
            "int" -> int_uops op
          | "float" -> float_uops op
          | "bool" -> bool_uops op
          | _ -> raise(UnsupportedUnop)
        in
        build_ops_with_type e'_type
      | A.Assign (e1, e2) -> let e1' = (match e1 with
                                            A.Id s -> lookup s
                                                                                | A.RowAccess(s,
e1) -> let i1 = expr builder e1 in build_row_access s (L.const_int i32_t 0) i1 builder true
                                                                                |
A.TupleAccess(s, e1) -> let i1 = expr builder e1 in build_tuple_access s (L.const_int i32_t
0) i1 builder true
                                                                                |
A.MatrixAccess(s, e1, e2) -> let i1 = expr builder e1 and i2 = expr builder e2 in
build_matrix_access s (L.const_int i32_t 0) i1 i2 builder true
                                                                                | A.MRowAccess(s,
e1) -> let i1 = expr builder e1 in build_mrow_access s (L.const_int i32_t 0) i1 builder true
                        | A.PointerIncrement(s) -> build_pointer_increment s builder true
                        | A.Dereference(s) -> build_pointer_dereference s builder true
                        | _ -> raise (IllegalAssignment))
                               and e2' = expr builder e2 in
                      ignore (L.build_store e2' e1' builder); e2'
      | A.Call("open", e) ->
    L.build_call open_func (Array.of_list (List.rev (List.map (expr builder) (List.rev e))))
"fopen" builder
      | A.Call("close", e) ->
    L.build_call close_func (Array.of_list (List.rev (List.map (expr builder) (List.rev
e)))) "fclose" builder
      | A.Call("write", e) ->
    L.build_call fwrite_func (Array.of_list (List.rev (List.map (expr builder) (List.rev
```

```
e)))) "tmpy" builder
      | A.Call("read", e) ->
    L.build_call read_func (Array.of_list (List.rev (List.map (expr builder) (List.rev e))))
"tmpx" builder
      | A.Call("fget", e) ->
    L.build_call get_func (Array.of_list (List.rev (List.map (expr builder) (List.rev e))))
"tmpz" builder
      | A.Call("len", e) ->
    L.build_call strlen_func (Array.of_list (List.rev (List.map (expr builder) (List.rev
e)))) "len" builder
      | A.Call ("atoi", e) ->
    L.build_call cast_str_int_func (Array.of_list (List.rev (List.map (expr builder)
(List.rev e)))) "tmp3" builder
      | A.Call ("itos", e) ->
    L.build_call sprintf_func (Array.of_list (List.rev (List.map (expr builder) (List.rev
e)))) "sprintf" builder
      | A.Call("splitstr", e) ->
     L.build_call strtok_fun (Array.of_list (List.rev (List.map (expr builder) (List.rev
e)))) "tmp5" builder
      | A.Call("find", e) ->
    L.build_call strfind_func (Array.of_list (List.rev (List.map (expr builder) (List.rev
e)))) "find" builder
      | A.Call("strcmp", e) ->
    L.build_call strcmp_func (Array.of_list (List.rev (List.map (expr builder) (List.rev
e)))) "strcmp" builder
      | A.Call ("print", [e]) | A.Call ("printb", [e]) ->
    L.build_call printf_func [| int_format_str ; (expr builder e) |]
      "printf" builder
      | A.Call ("printf", [e]) ->
    L.build_call printf_func [| float_format_str ; (expr builder e) |]
      "printf" builder
      | A.Call ("printfs", [e]) ->
    L.build_call prints_func [|(expr builder e)|] "puts" builder
      | A.Call("print_c",[e]) ->
    L.build_call printf_func [| char_format_str; (expr builder e)|]
      "printf" builder
      | A.Call("prints",[e]) -> let get_string = function A.StringLit s -> s | _ -> "" in
       let s_ptr = L.build_global_stringptr (get_string e) ".str" builder in
       L.build_call printf_func [| s_ptr |] "printf" builder
      | A.Call (f, act) ->
         let (fdef, fdecl) = StringMap.find f function_decls in
   let actuals = List.rev (List.map (expr builder) (List.rev act)) in
   let result = (match fdecl.A.typ with A.Void -> ""
                                        | _ -> f ^ "_result") in
         L.build_call fdef (Array.of_list actuals) result builder
    in

    (* Invoke "f builder" if the current block doesn't already
       have a terminal (e.g., a branch). *)
```

```
  let add_terminal builder f =
    match L.block_terminator (L.insertion_block builder) with
Some _ -> ()

    | None -> ignore (f builder) in

  (* Build the code for the given statement; return the builder for
     the statement's successor *)
  let rec stmt builder = function
      A.Block sl -> List.fold_left stmt builder sl
    | A.Expr e -> ignore (expr builder e); builder
    | A.Return e -> ignore (match fdecl.A.typ with
       A.Void -> L.build_ret_void builder
      | _ -> L.build_ret (expr builder e) builder); builder
      | A.If (predicate, then_stmt, else_stmt) ->
         let bool_val = expr builder predicate in
let merge_bb = L.append_block context "merge" the_function in

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

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

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

    | A.While (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

    | A.For (e1, e2, e3, body) -> stmt builder
    ( A.Block [A.Expr e1 ; A.While (e2, A.Block [body ; A.Expr e3]) ] )


       | A.MFor (s1, s2, body) ->
```

```
              let rows = L.array_length (L.type_of (L.build_load (L.build_gep (lookup s2)
[| L.const_int i32_t 0 |] s2 builder) s2 builder)) in
              let cols = L.array_length (L.type_of (L.build_load (L.build_gep (lookup s2)
[| L.const_int i32_t 0; L.const_int i32_t 0 |] s2 builder) s2 builder)) in
              allocate_row s1 cols s2;

              stmt builder
                ( A.Block
                     [A.Expr (A.Assign(A.Id "_i", A.IntLit 0));
                      A.While ((A.Binop((A.Id "_i"), A.Less, (A.IntLit rows))),
                        A.Block [A.Expr (A.Assign((A.Id s1), A.MRowAccess(s2, (A.Id
"_i")))); body ; A.Expr(A.Assign((A.Id "_i"), A.Binop((A.Id "_i"), A.Add, (A.IntLit 1))))])
] )
       in


    (* Build the code for each statement in the function *)
    let builder = stmt builder (A.Block fdecl.A.body) in

    (* Add a return if the last block falls off the end *)
    add_terminal builder (match fdecl.A.typ with
        A.Void -> L.build_ret_void
      | A.Int -> L.build_ret (L.const_int i32_t 0)
      | A.Float -> L.build_ret (L.const_float float_t 0.0)
      | A.Bool -> L.build_ret (L.const_int i1_t 0)
      | A.String -> L.build_ret (L.const_pointer_null (pointer_t i8_t))
      | A.MatrixPointer(t) -> (match t with
                                  A.Int -> L.build_ret (L.const_int (pointer_t i32_t) 0)
                                  | A.Float -> L.build_ret (L.const_float (pointer_t
float_t) 0.0)
                                  | _ -> raise (IllegalPointerType))
      | _ -> raise (UnsupportedReturnType))
  in

  List.iter build_function_body functions;
  the_module
```

g.

JSTEM.ml

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

type action = AST | LLVM_IR | Compile

let _ =
  let action = if Array.length Sys.argv > 1 then
    List.assoc Sys.argv.(1) [ ("-a", AST);  (* Print the AST only *)
```

```
            ("-l", LLVM_IR);  (* Generate LLVM, don't check *)
            ("-c", Compile) ] (* Generate, check LLVM IR *)
    else Compile in
    let lexbuf = Lexing.from_string (Preprocess.process_files Sys.argv.(2) Sys.argv.(3) )in
    let ast = Parser.program Scanner.token lexbuf in
    Semant.check ast;
    match action with
      AST -> print_string (Ast.string_of_program ast)
    | LLVM_IR -> print_string (Llvm.string_of_llmodule (Codegen.translate ast))
    | Compile -> let m = Codegen.translate ast in
      Llvm_analysis.assert_valid_module m;
      print_string (Llvm.string_of_llmodule m)
```

h. Makefile

Author: Sam Stultz

```
# Make sure ocamlbuild can find opam-managed packages: first run
#
# eval `opam config env`

# Easiest way to build: using ocamlbuild, which in turn uses ocamlfind

.PHONY : JSTEM.native
JSTEM.native :
	ocamlbuild -use-ocamlfind -pkgs
llvm,llvm.analysis,llvm.bitwriter,llvm.bitreader,llvm.linker,str -cflags -w,+a-4 \
		JSTEM.native

# "make clean" removes all generated files

.PHONY : clean
clean :
	ocamlbuild -clean
	rm -rf testall.log *.diff microc scanner.ml parser.ml parser.mli
	rm -rf *.cmx *.cmi *.cmo *.cmx *.o *.s *.ll *.out *.exe
	cd tests; rm -rf *.err *.diff *.ll *.o *.out; make clean

# More detailed: build using ocamlc/ocamlopt + ocamlfind to locate LLVM

OBJS = ast.cmx codegen.cmx parser.cmx scanner.cmx semant.cmx JSTEM.cmx

JSTEM : $(OBJS)
	ocamlfind ocamlopt -linkpkg -package llvm -package llvm.analysis -o JSTEM $(OBJS)

scanner.ml : scanner.mll
	ocamllex scanner.mll

parser.ml parser.mli : parser.mly
	ocamlyacc parser.mly
```

```
%.cmo : %.ml
	ocamlc -c $<

%.cmi : %.mli
	ocamlc -c $<

%.cmx : %.ml
	ocamlfind ocamlopt -c -package llvm $<


TESTOBJS = parser.cmo scanner.cmo

.PHONY : testbuild
testbuild : JSTEM.native test_parser_scanner

.PHONY : test_parser_scanner
test_parser_scanner : $(TESTOBJS)


test: testbuild
	cd tests; make

### Generated by "ocamldep *.ml *.mli" after building scanner.ml and parser.ml
ast.cmo :
ast.cmx :
codegen.cmo : ast.cmo
codegen.cmx : ast.cmx
microc.cmo : semant.cmo scanner.cmo parser.cmi codegen.cmo ast.cmo
microc.cmx : semant.cmx scanner.cmx parser.cmx codegen.cmx ast.cmx
parser.cmo : ast.cmo parser.cmi
parser.cmx : ast.cmx parser.cmi
scanner.cmo : parser.cmi
scanner.cmx : parser.cmx
semant.cmo : ast.cmo
semant.cmx : ast.cmx
parser.cmi : ast.cmo

# Building the tarball
TARFILES = ast.ml codegen.ml Makefile JSTEM.ml parser.mly README scanner.mll \
	semant.ml testall.sh $(TESTFILES:%=tests/%) printbig.c arcade-font.pbm \
	font2c

JSTEM-llvm.tar.gz : $(TARFILES)
	cd .. && tar czf JSTEM-llvm/JSTEM-llvm.tar.gz \
		$(TARFILES:%=JSTEM-llvm/%)
```

I. **stdlib.JSTEM**

Author: Julia Troxell

```
/*
NAME: print_tuple
ARGS: tuple pointer
RETURN: void
Desc: print a tuple
*/
def void print_tuple(int(%3%) t) {
	int i;
	prints("( ");
	for (i = 0; i < 3; i=i+1) {
		print(t(%i%));
		if (i < 2) {
			prints(", ");
		}
	}
	prints(" )");
}

/*
NAME: print_rowi
ARGS: int row pointer, int
RETURN: void
Desc: print an int row
*/
def void print_rowi(int[] r, int len) {
	int i;
	int len_minus_1;
	len_minus_1 = len - 1;
	prints("[ ");
	for (i = 0; i < len; i=i+1) {
		print(#r);
		if (i < len_minus_1) {
			prints(", ");
		}
		r = ~~r;
	}
	prints(" ]");
}

/*
NAME: print_rowf
ARGS: float row pointer, int
RETURN: void
Desc: print a float row
*/
def void print_rowf(float[] r, int len) {
```

```
	int i;
	int len_minus_1;
	len_minus_1 = len - 1;
	prints("[ ");
	for (i = 0; i < len; i=i+1) {
		printf(#r);
		if (i < len_minus_1) {
			prints(", ");
		}
		r = ~~r;
	}
	prints(" ]");
}

/*
NAME: print_rowt
ARGS: tuple row pointer, int
RETURN: void
Desc: print a tuple row
*/
def void print_rowt(int(%3%)[] r, int len) {
	int i;
	int len_minus_1;
	len_minus_1 = len - 1;
	prints("[ ");
	for (i = 0; i < len; i=i+1) {
		print_tuple(#r);
		if (i < len_minus_1) {
			prints(", ");
		}
		r = ~~r;
	}
	prints(" ]");
}

/*
NAME: print_matrixi
ARGS: int matrix pointer, int, int
RETURN: void
Desc: print an int matrix
*/
def void print_matrixi(int[[]] m, int len, int wid) {
	int i;
	int j;
	int len_minus_1;
	int wid_minus_1;
	len_minus_1 = len - 1;
	wid_minus_1 = wid - 1;
	prints("{ ");
```

```
	for (i = 0; i < len; i=i+1) {
		for (j = 0; j < wid; j=j+1) {
			print(#m);
			if (j < wid_minus_1) {
				prints(", ");
			}
			else if (i < len_minus_1) {
				prints("\n  ");
			}
			m = ~~m;
		}
	}
	prints(" }");
}

/*
NAME: print_matrixf
ARGS: float matrix pointer, int, int
RETURN: void
Desc: print a float matrix
*/
def void print_matrixf(float[[]] m, int len, int wid) {
	int i;
	int j;
	int len_minus_1;
	int wid_minus_1;
	len_minus_1 = len - 1;
	wid_minus_1 = wid - 1;
	prints("{ ");
	for (i = 0; i < len; i=i+1) {
		for (j = 0; j < wid; j=j+1) {
			printf(#m);
			if (j < wid_minus_1) {
				prints(", ");
			}
			else if (i < len_minus_1) {
				prints("\n  ");
			}
			m = ~~m;
		}
	}
	prints(" }");
}

/*
NAME: print_matrixt
ARGS: tuple matrix pointer, int, int
RETURN: void
Desc: print a tuple matrix
```

```
*/
def void print_matrixt(int(%3%)[[]] m, int len, int wid) {
	int i;
	int j;
	int len_minus_1;
	int wid_minus_1;
	len_minus_1 = len - 1;
	wid_minus_1 = wid - 1;
	prints("{ ");
	for (i = 0; i < len; i=i+1) {
		for (j = 0; j < wid; j=j+1) {
			print_tuple(#m);
			if (j < wid_minus_1) {
				prints(", ");
			}
			else if (i < len_minus_1) {
				prints("\n  ");
			}
			m = ~~m;
		}
	}
	prints(" }");
}

/*
NAME: add_rowi
ARGS: int matrix pointer, int matrix pointer, int row pointer, int, int
RETURN: int matrix pointer
Desc: adds a row to an int matrix, returns a pointer to a new matrix
*/
def int[[]] add_rowi(int[[]] oldm, int[[]] newm_ptr, int[] row, int len, int wid) {
	int i;
	int j;
	int len_plus_1;
	int offset1;
	int offset;
	int[[]] newm;

	offset1 = len * wid;
	offset = offset1 + wid;
	len_plus_1 = len + 1;
	newm = newm_ptr;

	for(i = 0; i < offset; i=i+1) {
		if (i < offset1) {
			#newm = #oldm;
			oldm = ~~oldm;
			newm = ~~newm;
		}
```

```
		else {
			#newm = #row;
			row = ~~row;
			newm = ~~newm;
		}
	}
	return newm_ptr;
}

/*
NAME: add_rowf
ARGS: float matrix pointer, float matrix pointer, float row pointer, int, int
RETURN: float matrix pointer
Desc: adds a row to a float matrix, returns a pointer to a new matrix
*/
def float[[]] add_rowf(float[[]] oldm, float[[]] newm_ptr, float[] row, int len, int wid) {
	int i;
	int j;
	int len_plus_1;
	int offset1;
	int offset;
	float[[]] newm;

	offset1 = len * wid;
	offset = offset1 + wid;
	len_plus_1 = len + 1;
	newm = newm_ptr;

	for(i = 0; i < offset; i=i+1) {
		if (i < offset1) {
			#newm = #oldm;
			oldm = ~~oldm;
			newm = ~~newm;
		}
		else {
			#newm = #row;
			row = ~~row;
			newm = ~~newm;
		}
	}
	return newm_ptr;
}
```

J. **demo.JSTEM**

```
def int main() {

	/* Declare variables */
```

```
int[3][3] m;
int[3][3] m1;
int[3][3] m2;
int[3][3] m3;
float[2][2] m4;
float[3][2] m5;
float[2] r1;

int len;
int len_plus_1;
int wid;
int i;
int j;

String line;
String temp;
String delim1;
String inttemp;

String null;

File file_in;
File file_out;

line @= new[1000];
temp @= new[1000];
inttemp @= new[3];

delim1 @= new[1];
delim1 = ",";

null = find("", "x");

file_in = open("matrix.txt","r");

fget(line, 10, file_in);
temp = splitstr(line,delim1);
m[0][0] = atoi(temp);


for (i = 1; i < 3; i = i + 1){
        m[0][i] = atoi(splitstr(null,delim1));
}

fget(line, 1000, file_in);
temp = splitstr(line,delim1);
m[1][0] = atoi(temp);

for (i = 1; i < 3; i = i + 1){
```

```
                m[1][i] = atoi(splitstr(null,delim1));
        }

        fget(line, 1000, file_in);
        temp = splitstr(line,delim1);
        m[2][0] = atoi(temp);

        for (i = 1; i < 3; i = i + 1){
                m[2][i] = atoi(splitstr(null,delim1));
        }



        /* Demonstrate matrix arithmetic, matrix scalars, & misc functions in standard
library */

        m1 = m * 3;
        prints("m1\n");
        len = 3;
        wid = 3;
        print_matrixi($$m1,len,wid);
        prints("\n");

        m2 = {% 9, 8, 7 | 6, 5, 4 | 3, 2, 1 %};
        prints("m2\n");
        len = 3;
        wid = 3;
        print_matrixi($$m2,len,wid);
        prints("\n");

        /* adds matrix 1 and matrix 2 to result matrix 3 */
        m3 = m1 + m2;
        prints("m3 = m1 + m2\n");
        len = 3;
        wid = 3;
        print_matrixi($$m3,len,wid);
        prints("\n");

        /* multiples matrix 1 by scalar 2 to result matrix 3 */
        m3 = m1 * 2;
        prints("m3 = m1 * 2\n");
        print_matrixi($$m3,len,wid);
        prints("\n");

        m4 = {% 1.2, 3.5 | 0.7, 4.9 %};
        prints("m4\n");
        len = 2;
        wid = 2;
        print_matrixf($$m4,len,wid);
```

```
        prints("\n");

        r1 = [ 0.5, 0.5 ];
        len = 2;
        len_plus_1 = len + 1;
        prints("r1\n");
        print_rowf($r1,len);
        prints("\n");

        /* appends row to matrix 4 to result matrix 5 */
        prints("Append r1 to m4 to get m5\n");
        add_rowf($$m4,$$m5,$r1,len,wid);
        prints("m5\n");
        print_matrixf($$m5,len_plus_1,wid);
        prints("\n");


        file_out = open("matrix_out.txt", "w");

        for (i = 0; i < 3; i = i + 1){
                for (j = 0; j < 3; j = j + 1){

                        itos(inttemp, "%d", m3[i][j]);
                        write(inttemp, 1, len(inttemp), file_out);

                        if (j != 3) {
                                write(",", 1, 1, file_out);
                        }
                }
                write("\r\n", 1, 2, file_out);
        }

        close(file_in);
        close(file_out);

        return 0;
}
```

matrix.txt

```
1,2,3
4,5,6
7,8,9
```

matrix_out.txt

```
6,12,18,
```

```
24,30,36,
42,48,54,
```

demo.JSTEM output

```
m1
{ 3, 6, 9
  15, 18, 21
  27, 30, 33 }
m2
{ 9, 8, 7
  6, 5, 4
  3, 2, 1 }
m3 = m1 + m2
{ 12, 14, 16
  21, 23, 25
  30, 32, 34 }
m3 = m1 * 2
{ 6, 12, 18
  30, 36, 42
  54, 60, 66 }
m4
{ 1.200000, 3.500000
  0.700000, 4.900000 }
r1
[ 0.500000, 0.500000 ]
Append r1 to m4 to get m5
m5
{ 1.200000, 3.500000
  0.700000, 4.900000
  0.500000, 0.500000 }
```

K. demo1.JSTEM

```
def int main() {
        int(%3%)[2][2] m1;
        int len;

        m1 = {% (%1, 1, 1%), (%2, 2, 2%) | (%3, 3, 3%), (%4, 4, 4%) %};
        len = m1.length;

        for(any_variable_name in m1){
                print_rowt($any_variable_name, len);
                prints("\n");
        }

        return 0;
}
```

demo1.JSTEM output

```
[ ( 1, 1, 1 ), ( 2, 2, 2 ) ]
[ ( 3, 3, 3 ), ( 4, 4, 4 ) ]
```