Stitch Language Final Report

Daniel Cole (System Architect), Megan Skrypek (Tester), Rashedul Haydar (Manager), Tim Waterman (Language Guru)
dhc2131, ms4985, rh2712, tbw2105

December 22, 2015

Introduction

Most "modern" programming languages trace their origins back decades to before the advent of cheap, general purpose multicore CPUs. They were designed for a distinctly mono-threaded environment. While libraries and enhancements to mainstay languages such as C/C++ and Java have added multithreading capabilities, it remains in many ways bolted on kludge. While newer frameworks such as Node.js provide more integrated support for asynchronous operations, they lack the depth of support and power of a fully compiled language. With Stitch, we aim to build a language that has the power and flexibility of a fully compiled C style language, while having native threading support for modern multithreaded applications. Our goal was to create a translator from Stitch to C.

Stitch is inspired by C, which has a very well known syntax, and has been one of the most widely used languages since it was released over forty years ago. Stitch is a general purpose language that supports all standard mathematical and logical operations. Like C, Stitch is strongly typed, and whitespace does not matter. Stitch supports the standard C primitive types `int, double, char`.

Stitch is able to provide an easy to use, clear paradigm for multithreaded operations by strictly limiting when and how they can be invoked. This is done through the `stitch` loop. The body of this loop is automatically split into multiple threads, and the program will not continue until all threads have returned. Using a simple loop paradigm, similar to well known control structures like `while` and `for` loops, allows for an easy learning curve, and clear easy to read code. It also allows the complier to easily see what code needs to be run in a threaded manner, and to efficiently generate the threaded code.

The underlying method by which Stitch runs multithreaded code is C's `pthread` library. The Stitch complier will wrap the body of the `stitch` loop in a function. This function will be executed in parallel using `pthreads`. Variable scoping inside the threads is also handled by the complier. Each thread is passed a C `struct` that contains all non-local variables needed by the block of code that is being multithreaded. This prevents clobbering issues without needing to resort to `mutex` locks. The only exceptions to this rule are accumulators, which are very limited in scope, and arrays, which can be sliced and piecewise accessed by different threads concurrently.

Language Tutorial

Running The Stitch Compiler:

When inside the ocaml folder, type `$ make all` in order to generate the stitch executable. Running `$ ./singer filename.stch` from the home directory will output a C program called filename.stch.c which gets compiled in singer with the appropriate C libraries and runtime headers into an executable of the same file name. Singer needs to be in the home directory in order to access the compiler executable and runtime headers correctly, if it needs to be moved then those directory accesses need to be updated. The file being compiled by `singer` also needs to be in the home directory.

Hello World

This is the popular "hello world" program written in Stitch. As can be seen below, it's almost identical to how it would be written in C, except without the #include statement and the syntax of the print function.

```
1 int main()
2 {
3   print("hello world");
4   return 0;
5 }
```

Matrix multiplication

If you want to use the multithreaded feature of Stitch, then simply use the stitch loop. Matrix multiplication is shown using the stitch loop below.

```
int main() {

  int a[5][5] = { {1, 2, 3, 4, 5},
          {1, 2, 3, 4, 5},
          {1, 2, 3, 4, 5},
          {1, 2, 3, 4, 5},
          {1, 2, 3, 4, 5} };

  int b[5][5] = { {1, 1, 1, 1, 1},
          {2, 2, 2, 2, 2},
          {3, 3, 3, 3, 3},
          {4, 4, 4, 4, 4},
          {5, 5, 5, 5, 5} };

  int c[5][5];

  int i = 0;
  int j = 0;
  int k = 0;

  stitch i from 0 to 5 by 1: {

    for(j = 0; j < 5; j = j + 1) {

      for(k = 0; k < 5; k = k + 1) {

        c[i][j] = c[i][j] + a[i][k] * b[k][j];
      }
    }
  }

  for(j = 0; j < 5; j = j + 1) {

    for(k = 0; k < 5; k = k + 1) {

      print(c[j][k]);
    }
  }


  return 0;
}
```

Language Reference Manual

1 Types

1.1 Primitive Data Types

Stitch supports a number of primitive data types: integers, characters, and floating point numbers.

1.1.1 Numeric Data Types

Stitch has support for two basic numeric data types, `int` and `float`.

- `int`
  Integers are 32-bit signed fixed precision numbers.

- `float`
  Floats are single precision floating points.

1.1.2 Accumulators

In addition to basic numeric data types, there also exists one numeric data type for accumulators that are to be used inside the Stitch loops. It is:

- `int_ap`

It is equivalent to its counterpart, `int`, in the sense that it could potentially be used outside Stitch loops, and would behave as a normal `int`. However, this usage is discouraged to prevent confusion on which variables are accumulators and which ones are regular numerical data types. The `_ap` are abbreviation is for additive (plus) accumulator (`_ap`). At the moment, accumulators are limited to arrays of size 4.

1.1.3 Characters

Chars in Stitch are exactly the same as their C counterparts; they are one byte variables that hold a value representative of an alphanumeric character or punctuation.

1.1.4 Arrays

An array is a data structure that lets you store one or more elements consecutively in memory.

Arrays can store any of the numerical or character data types (float, int, char).

There are two ways to declare an array:

```
<type> arrayName[size];
<type> arrayName[size] = {value-0,value-1,...,value-(size-1)};
```

The first declaration creates an array of size `size`, which has to be an int literal, and the values of the cells are undefined until you manually change them. The second declaration will initialize an array with the values passed to it, and the length of the set of initial arguments must match the size of the array.

You can declare an array with either the [size] by itself or with the {initial elems}. So the following are invalid array declarations in Stitch:

```
<type> arrayName[];
<type> arrayName[] = {value-0,value-1,value-2};
```

To access an element of an array, you use C-style square bracket notation:

```
arrayName[index]
```

1.1.5 Matrices

Matrices are two-dimensional arrays, and are declared in a very similar fashion to their one-dimensional counterparts:

```
<type> arrayname[numRows, numCols];
```

This will create an array of type `<type>` with a total number of elements equal to `numRows * numCols`.

The size parameters are also not optional, and must match the dimensions of the initialized matrix.

```
int d[3][3] = { {2,3,1}, {4,6,5} };
```

This will create a 2D array of ints named d, whose first row is {2,3,1} and whose second row is {4,6,5}.

If the size parameters are included, but the number of elements initialized does not match, this is invalid behavior and will not compile.
An example:

```
float array m[4][4] = { {1,2,3,4}, {5,6}, {7,8,9} };
```

Will not work.

Stitch will not catch array bounds exceptions at compile time, but at runtime.

1.2 String Literals

Stitch will support string literals. String literals cannot be assigned to a variable. However, they can be used inside `print`(), `error()` and file I/O statements.

1.3 Casting

Stitch does not support casting of any of its data types. Therefore, for any binary operators, the types of the operands must match.

2 Lexical Conventions

2.1 Declarations and Identifiers

A declaration in Stitch associates an identifier with a stitch object. Variables and functions may be so named. The name of a declared identifier in Stitch must begin with an alphabetic character (unlike C, a leading underscore is not permitted), and may contain any further number of alphanumeric characters and underscores. Stitch does not support characters other than [‘0’-’9’ ‘a’-’z’ ‘A’-’Z’ ‘_’] in valid declarable names.

2.2 Literals

- char literals

    - For all common ASCII characters a literal is expressed as the character surrounded by single quotes.
    - Characters that require escaping, because they have no equivalent typable glyph, or because they have special meaning are escaped by a backslash, and then surrounded by single quotes. The following characters must be escaped as such:
        - `'\\'` - backslash
        - `'\''` - single quote
        - `'\"'` - double quote
        - `'\n'` - newline
        - `'\t'` - tab

- int literals
    - one or more digits without a decimal point, and with an optional sign component

- float literals
    - one or more digits with a decimal point, and with an optional sign component

For both `int` and `float` literals, the maximum representable value is determined by the underlying C implementation.

- array literal
    - an array literal is a comma separated list of literals enclosed by curly braces. Multidimensional arrays are made by nesting arrays within arrays.
- string literal
    - a string literal is a sequence of one or more chars, enclosed by double quotes.

2.3 Whitespace

In Stitch, whitespace consists of the space, tab, and newline characters. Whitespace is used for token delimitation, and has no other syntactic meaning.

2.4 Comments

In Stitch, as in C, single line comments are delimited by the double forward slash characters. Multiline comments begin with the forward slash character, followed by the asterisk character. They continue until they are ended by an asterisk followed by a forward slash.

2.5 Punctuation

- single quote - `‘`
    - used to encapsulate a char literal
- double quote - `“`
    - used to encapsulate string literals
- parentheses - `()`
    - function arguments
    - conditional predicates
    - expression precedence
- square brackets - `[]`
    - array access
    - array declaration
- curly braces - `{}`
    - array declaration, function definitions, block statements
- comma - `,`
    - function parameter separation
    - array literal separation
- semicolon - `;`
    - end of statement
- colon - :
    - end of Stitch declaration

2.6 Operators

Stitch includes a simplified subset of the C operators, including all basic arithmetic operators. All operators may be used freely in stitch loops.

Arithmetic Operators:

`*`	`Multiplication`
`/`	`Division`

+	Addition
-	Subtraction
%	Mod

Assignment, Negation, and Equivalence Operators:

=	Assignment
==	Equivalence
!	Negation
!=	Non-Equivalence

Logical Operators:

&&	Logical AND
||	Logical OR

Comparison Operators:

>	Greater Than
<	Less Than
>=	Greater Than or Equal To
=<	Less Than or Equal To

2.7 Operator Precedence

In Stitch, arithmetic operator precedence will follow standard arithmetic conventions. Comparison operators have precedence as in C.

2.8 Keywords

- `if(condition)`
- `else`

- `while(condition)`
- `for(assignment; condition; expression)`
- `stitch` *`variable`* `from` *`startRange`* `to` *`endRange`* `by` *`stepsize`* `:`
- `break`
- `return`
- `void`
- `main(expression, expression)`

3 Stitch Loops & Multi-threading

A key feature in Stitch is the inclusion of multithreading on certain loop constructs. When you use these loops, the body of the loop will be split into separate threads and run concurrently. The splitting, thread management, and cleanup are all handled by the compiler. The loops are called stitch loops, and can be called using the following syntax:

```
stitch variable from startRange to endRange by stepsize :
```

Variable is a counter variable that must be an integer which must be declared before the loop. startRange and endRange are either numeric literals or expressions that evaluate to numeric literals. The variable will begin at the value of `startRange` and increment by the value of stepsize (which is a signed integer value) until the value of `endRange`. In keeping with traditional C paradigms, the range represented by `startRange,endRange` is [startRange,endRange). That is, it is inclusive on the start but exclusive on the end. What follows is an example of a typical C-style for loop with an equivalent stitch loop.

```
for(i = 0; i < 10; i++)

stitch i from 0 to 10 by 1 :
```

The body of the for loop will then be executed in parallel while the main program thread blocks and waits for the threads to return. The variable, while it can be used as an index

to access the current iteration, can never be assigned to; that is, it cannot be an **lvalue** inside a loop of this structure where it is used as an assignment. Vector operations are not allowed inside asynchronous loops, and so having vector operations in a stitch loop will result in compilation errors.

4 Syntax

4.1 Program Structure

The overall syntax of Stitch is very similar to C's syntax, with some minor differences, especially when it comes to the asynchronous parts of the program. The general structure of the program will contain a `main()` function. When the program executes, the body of the `main()` function will be executed along with any functions defined outside of the `main()` function. All other statements will not be run.

Variables cannot be declared outside of the `main()` function, thus global variables do not exist in the Stitch language. Also, since there is no concept of pointers in Stitch, the generic structure of the `main()` function in C

```
int main(int argc, char **argv)
```

would not work because of the `char **`. However, normal formal arguments still work, such as the `int argc` component above, but they aren't useful for main because Stitch has no stdin.

4.2 Expressions

Expressions in Stitch have a type and value associated with them, and consist of operators and operands. The order of evaluation of the expressions is from left to right, unless there are parentheses, in which case the expression inside the innermost parentheses gets evaluated first.

4.2.1 Assignment

Assignment is done using the `'='` symbol. The value of the expression on the right hand side is stored in the variable on the left hand side. The syntax for assignment is as follows:

```
variable = value;
```

```
arrayName[index] = value;
```

4.2.2 Arithmetic

Arithmetic operators are plus `+`, minus `-`, multiplication `*`, division `/`, and modulus `%`. The of arithmetic operators can only be expressions of type int or float. The evaluated value is of the same type. For the `+` and `-` operators, there must be spaces between the operands and the operator. The syntax for the plus operator is shown below for guidance. The same is not true for the rest of the binary operators. Because of this, it's highly suggested that there be spaces for all binary operators, not just addition and subtraction, for consistency.

```
operand1 + operand2
```

4.2.3 Comparison

Comparison operators are less-than-or-equal-to `<=`, less-than `<`, greater-than `>`, greater-than-or-equal-to `>=`, equal-to `==`, and not-equal-to `!=`. The operands can be of any type, but must match. It is not possible to compare `int`s and `float`s, for example. The return type of a comparison is always int, and the value returned is either 0 (false) or nonzero (true).

Stitch only supports comparison on primitive data types. Therefore, comparison on arrays is not possible.

```
arrayName1 == arrayName2;           //syntax error
```

4.2.4 Logical

Logical operators are AND `&&`, and OR `||`. The operands of logical operators must have type int, and the return value is of type int and has values 0 or 1.

4.3 Statements

A statement in Stitch is a full instruction, the end of which must be denoted by a semicolon `;`. Multiple statements can be encapsulated by `{` and `}`, and becomes a block.

4.3.1 Conditional Statements

Conditional statements use the `if` and `else` keywords and express decisions. The syntax is as follows:

```
if(expression)
      statement1
else
      statement2
```

If the `expression` evaluates to an integer >0, then `statement1` executes, otherwise `statement2` would execute.

Alternatively, for multiple decisions there can be `else if` blocks, the same as C. The syntax for that is:

```
if(expression1)
      statement1
else if(expression2)
      statement2
else
      statement3
```

In this situation, if `expression1` evaluates to >0, then `statement1` would execute, and the rest of the `else if` and `else` blocks are terminated. The expressions are evaluated in order. The last `else` is optional, and in general, an `else` always attaches itself to the preceding `else`-less `if`.

4.3.2 Loops

There are three types of loops in Stitch: `for`, `while`, and `stitch` loops. The `for` and `while` loops have the same structure as in C, but the `stitch` loop has a different syntax. The following shows how to use the `stitch` loop.

```
stitch variable from startRange to endRange by stepsize: statement
```

Further explanation of the `stitch` loop is provided in section 4.

4.3.3 Loop Disruptions

The keyword `break` can be used inside of all three types of loops. It will cause the innermost loop containing the `break` statement to terminate.

4.3.4 Returns

The keyword `return` is used to return the value of an expression from a function to the caller. Anything after the `return` statement is not executed. Every non-void function, including `main`, must have a return of the proper type.

4.3.5 Functions

A function statement calls a function and returns a value if the called function has a `return` statement. The return type must be present for a function declaration. If nothing is to be returned from the function, then the return type should be `void`. The syntax for a function definition is the following:

```
returnType functionName(formal_argument1, formal_argument2, ...)
{
      statements
      optional return statement
}
```

5 Standard Library Functions

Stitch provides a relatively small number of standard library functions. These are used to facilitate I/O, and as a convenience to facilitate basic operations.

5.1 I/O Functions

Stitch provides the following functions for both file I/O and user I/O. These are drastically simplified versions of their C counterparts. Files are referenced by their file descriptor, which is stored as an integer value.

- `int write(File, array)` - write the data held in `array` to the file specified by `File`. Returns the number of elements written. Warning: if the file is not empty, fwrite() will overwrite some or all of the data stored in the file.

- `int read(File, array)` - read data from the file specified by `File` into the `array`. If there is more data in the file than can be stored in the array, the array will be filled, and and the read will stop. Returns the number of elements read.

- `FILE open_r(string_literal)` - opens a file for reading at the path specified in the `string_literal`. The file is opened in "r+" mode behind the scene in C. Returns a file descriptor.

- `FILE open_w(string_literal)` - opens a file for writing at the path specified in the `stirng_literal`. The file is opened in "w+" mode behind the scene in C. Returns a file descriptor. Calling both `open_r()` and `open_w()` on the same file name is undefined.

- `void print(expression)` - prints the specified `expression` to stdout. Functions cannot be called from within the `print()` function.

- `void error(expression)` - prints the specified `expression` to stderr.

5.2 Miscellaneous Functions

Stitch also provides the `exit()` function meant to aid the programmer.

- `exit(int)` - if called from the main body of the program, this exits the program with a code of int. If called in a stitch loop, `exit()` will exit all threads, as well as the main program. A wrapper for the C function exit().

Project Plan

Planning

We arranged weekly meetings with our language advisor Professor Edwards to discuss progress and issues that we encountered. The immediate feedback that was received from him was extremely helpful in the development of the language, especially when we were heading in the wrong direction. We had weekly meetings as well where all of us got together and worked on the project. During the meetings we split up the work, often two people working together on the same thing. Initially this worked really well since all of us were new with OCaml. From Thanksgiving on, we met multiple times a week, eventually forgetting the sweet embrace of sleep as we pushed on to finish the language.

Style Guide

While programming our compiler we tried to follow these general guidelines:

- Ocaml style guidelines, such as indentation and formatting
- Tried to keep lines limited to 80 characters, if this wasn't possible due to unreadability, then we used 120 characters as the hard limit.
- Unlike Ocaml, we named variables in all lowercase and used underscores as a delimiter
- Used 4-space indentation for each program

Project Timeline

September 30 Proposal submitted

October 26 LRM submitted, scanner and parser with 1 shift/reduce error

November 16 Working scanner, parser, ast without arrays/stitch loops, 'Hello, Word' works

November 30 Finished initial semantic analyzer and CAST

December 8 Finished C code generator with arrays added

December 16 Stitch loops working

December 21 Final Presentation

December 22 Code cleanup and Final Report submitted

Team Roles and Responsibilities

Rashedul Haydar	-	Manager
Tim Waterman	-	Language Guru
Dan Cole	-	System Architect
Megan Skrypek	-	Tester

While we had assigned roles, the responsibilities became much more fluid as the project progressed. During the initial planning phase we all discussed the structure and components of the language. In the final stages of the project, Dan and Tim worked on the semantic analyzer and the C generator components, while Megan and Rashedul worked on the tests used for the test suite and finalizing the LRM and the final report. After the initial, non-semantic 'Hello World', Dan wrote most of the initial semantic analyzer as well as initial work on the C Generator, drawing from the work done on the pretty printer in the AST. Tim added pretty much everything having to do with arrays, Dan took care of built in functions, as well as the initial stitch loops, including the generation of functions from the Stitch loop body. Tim did all the pthread code generation and stitch loop generation, with Dan helping a bit with the architecture of collecting and storing the stitch local variables.

Software Development Environment

- Version Control
  - Git
- Languages
  - OCaml (4.02.3) for parser, scanner, ast, semantic analysis
  - GCC for compiling generated C code
  - bash for test suite and singer
  - Python (2.7.5) for image curve generator
  - LaTeX for reports and documentation
- Text Editors
  - VIM
  - Sublime

Project Log

```
commit 14fcbbf36913eeba32454705951f590629df1ae1
Merge: f86f952 cd73bf1
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Mon Dec 21 12:27:34 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit f86f952f4ed913c92bc2b3d96d028ec7d1b554e7
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Mon Dec 21 12:27:29 2015 -0500

    updated demo and presentation

commit cd73bf1546f50a0d761cd73c237e5adc6cd58935
Author: Tim Waterman <watermantium@gmail.com>
Date:   Mon Dec 21 12:14:34 2015 -0500

    Getting everything up pre-demo

commit 7726285bd6845f42de94c180d981660829932ac7
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Mon Dec 21 12:11:17 2015 -0500

    updated presentation and demo

commit 219855f53fdf111f06eb251c678126fe991525fe
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Mon Dec 21 12:08:27 2015 -0500

    updated presntation

commit e34dfe37af0d6b8e38d138b08e95f0495c3ced2f
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Mon Dec 21 11:09:20 2015 -0500

    generated C for matrix multiplication added
```

```
commit e28346bb1af89c7959e2ea6309c3d4922651b9fc
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Mon Dec 21 10:56:32 2015 -0500

    added matrix multiplication code

commit 6b9558e962acaa2f7a2ceb2dc585a72fce937bcb
Author: Tim Waterman <watermantium@gmail.com>
Date:   Mon Dec 21 02:13:42 2015 -0500

    Got demo to work with no C code cheats

commit 833ab002702eb5b56b652087165fb424dbae01cb
Author: Tim Waterman <watermantium@gmail.com>
Date:   Mon Dec 21 01:27:49 2015 -0500

    Starting testing with + accumulators

commit b8434f0eb623b9ffee29834955d6df9bd966c057
Author: Tim Waterman <watermantium@gmail.com>
Date:   Mon Dec 21 00:14:15 2015 -0500

    Stitch loop scoping issues worked out

commit 5c465f254dd3266f7dbe482ae6c451ccc9524c67
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 23:57:31 2015 -0500

    Made a better matrix mult test; updated sems

commit 6d19c76c540b35ba8814c9f3b703fa7d1d21a485
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 21:50:28 2015 -0500

    Added matmult output

commit 55d411d2608ceb6180e82de2e2401e7c34a48a90
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 21:48:56 2015 -0500
```

```
    Matrix multiplication test working

commit 7bd1175b3e6aa95bc47918a6811abbdea525ce94
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 21:34:36 2015 -0500

    Working on scoping issues

commit b1ba4625bbd4613eefc6ed4871e364a22cf7e29c
Merge: a978103 5a49906
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 20:35:05 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit a9781034f1f1dd44de3ac97fe08fcf570818e847
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 20:35:00 2015 -0500

    added open_r and open_w

commit 5a49906dd846b14931a3e2c4aef86a73a58b0590
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 20:32:14 2015 -0500

    Finished matrix init -> stitch

commit c909820ba855fce2ca8a62a94dd186aa51d80197
Merge: 62cbf0a eca869c
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 20:19:10 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit eca869cb0f88e9615b4e4cdad106c259a07fd181
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 20:19:04 2015 -0500

    Fixed 1D array init passing
```

```
commit 62cbf0a8d1c68cbbd8a7c67a1ce79ce00330a8c1
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 20:19:02 2015 -0500

    minor fix

commit 6455582db726d6f5dd7e2dc7df0a96c0031e9b53
Merge: 3d2ceff c16ede4
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 19:57:53 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit 3d2ceffe122c54c3b87f5d878060b7644319a2cb
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 19:57:51 2015 -0500

    Finished stch_stmt checking

commit c16ede42f887a33020826c38f19451e1a7a8e96c
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 19:56:27 2015 -0500

    fixed paren issues

commit 1d273b474db7be10ec9c9ec3095e752ab28149a3
Merge: 4c6e669 8bd704e
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 19:50:10 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit 4c6e669fa31ea9667d8a96b9670393b6c4e63fed
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 19:50:08 2015 -0500

    Added some more generator testing
```

```
commit 8bd704ed2da598867247fe681028b4bc72469083
Author: ms4985 <ms4985@columbia.edu>
Date:   Sun Dec 20 19:43:46 2015 -0500

    removed func3

commit 39eafcbec26b5abdf473b24a59bc6f7e56ed34bc
Merge: 78c76df 03244dd
Author: ms4985 <ms4985@columbia.edu>
Date:   Sun Dec 20 19:42:43 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit 78c76dfd0801089d8f41cb25fb7708bd5d824ffb
Merge: 38d630a d30b9bf
Author: ms4985 <ms4985@columbia.edu>
Date:   Sun Dec 20 19:42:36 2015 -0500

    removed fib1

commit 03244dde2357d2cf0763b0c043d3ae3f7d54e9ec
Merge: 02836f8 d30b9bf
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 19:41:43 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit 02836f8476fb79c41b6b85fba37f56e69919eae5
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 19:41:41 2015 -0500

    Fixed array passing bug

commit 38d630a36e0cbbce22473bd3a30475b78f4e98c5
Author: ms4985 <ms4985@columbia.edu>
Date:   Sun Dec 20 19:41:21 2015 -0500

    added tests
```

```
commit d30b9bf0e270d8143e2113eaefbd791ebbd8ef2c
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 19:39:52 2015 -0500

    minor fix on 4

commit f515cf5f229418bef8d9530c1f65ef907ad88c38
Merge: 7fb08e9 200acf2
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 19:27:28 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
    _lang

commit 7fb08e934101b3837ee3d105ac06d42357ae54a9
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 19:27:18 2015 -0500

    2D arrays are working

commit 200acf257512624f312e6bd58b5b74ccf6a4d1ee
Merge: c36a877 aaae5ce
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 19:07:28 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
    _lang

commit c36a8772a17c1a1774704f8b9b2aa472199d955a
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 19:07:21 2015 -0500

    recursion wont work

commit aaae5ce4829084dcf6751e96cfd8360031d7a5e2
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Sun Dec 20 18:38:27 2015 -0500

    added _ntests/array3.stch, array4.stch, and matrixinit2.stch

commit 9957b7448d9e6fa3427d38785ee8a6a7da5b6634
```

```
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Sun Dec 20 17:47:41 2015 -0500

    added _tests/fib1.stch

commit 9847733e91ccd9d62bd444f8085ba6821f5b057a
Merge: 588cc44 515be64
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 17:43:42 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit 588cc44363e5e2809436a3b4baf4dfa90efe1f39
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 17:43:39 2015 -0500

    fixed recursion

commit 515be640f660365dd4ef70a62865a9f9bb9e7322
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 17:43:14 2015 -0500

    Can print locals in stitch loops

commit 0fcd77f0c164c463a99f1166662419b806c35141
Merge: 89b2f31 aceffe8
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Sun Dec 20 17:36:47 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit 89b2f31208a2116d9b3af94c7da2537a77580f42
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Sun Dec 20 17:36:34 2015 -0500

    added _/tests/func4 and func5.stch

commit aceffe8260737f1a83e5ce8c40f0efda6fe3fbe1
Merge: 02d3734 3c88e1c
```

```
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 17:34:58 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit 02d37343356e0b50624e7996a9b83e2d3d961eda
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 17:34:51 2015 -0500

    More code gen

commit 3c88e1c76fa14e81cc087364f249c83e50143ace
Author: ms4985 <ms4985@columbia.edu>
Date:   Sun Dec 20 17:34:19 2015 -0500

    renamed gcd added stitch4 output txt

commit 8c3fbe3a7588c1715967ac2e278e3273988016f9
Merge: 5f94a5a fa7c23e
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 17:23:19 2015 -0500

    Fixing the merge

commit 5f94a5a12f071674e2943f908bbc0d6f2ef3cf55
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 17:21:18 2015 -0500

    Stitch statement parsing overhaul

commit fa7c23ed6dbfc50cca96c6bc445591c2fd7d9f43
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 16:58:46 2015 -0500

    fixed .gitignore; char can now cast to int

commit 2e19c6d01c184a93898a82205a152202a2018bdd
Author: ms4985 <ms4985@columbia.edu>
Date:   Sun Dec 20 16:55:05 2015 -0500
```

```
    fixed escaped characters

commit 50a5d3ee12a1a96f4d5db00ea87fa9e26017f428
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Sun Dec 20 16:49:34 2015 -0500

    updated all tests to have correct return statements

commit 0354d0c3b716d48a1d5d6eea3f619d12e726c6ea
Merge: 7413e98 3fe7525
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 16:33:15 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit 7413e982270eb9db48be5ca875d06a15445df65b
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 16:32:57 2015 -0500

    check for return on non-void function

commit 3fe7525d89f8244769ddcc90b9679dc962286068
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Sun Dec 20 16:01:57 2015 -0500

    added _ntests/vardecl1.stch, cant identifier starting with _
   or a number

commit 1446057a4c0c33c04434dd7ca02b9257908a0a92
Merge: 29c3e59 1895079
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 15:33:05 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit 29c3e594d2e8672ea9c39be374bb5732efdd314b
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 15:32:59 2015 -0500
```

```
    Fixed issue with var being removed too late from stitch

commit 1895079fa4b0a460c0d2166d24d01f3984784322
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Sun Dec 20 15:25:49 2015 -0500

    added _tests/for1.stch

commit 350b17670a0b795902fc008094bc429d8aadf3b3
Merge: c79809b 1c46608
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 15:15:51 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
    _lang

commit c79809b7070dc507bd5c00756504e8cddcf12562
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 15:15:46 2015 -0500

    Adding gen stuff

commit 1c46608678604b7a6c09626205e00d62ee22f28c
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 15:05:06 2015 -0500

    fixed function ordering problem

commit ccae33c4a21f924abac64ed695b8e6298581147f
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Dec 20 14:59:01 2015 -0500

    Working on stitch loop verification

commit c447d69eb709f5becdfff2f0d500e609603df306
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 14:39:08 2015 -0500

    updated presentation and demo

commit 492a1ce0c48c369b9681892acd891d6bb950395e
```

```
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 13:56:02 2015 -0500

    added presentation pdf

commit 655acdd240fe5ef42ebb6b6f3859f9f9807d0b6a
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 13:55:24 2015 -0500

    updated file2 test

commit 3954666f247f91a72988b0b63b457b63a66dbce9
Merge: 6a93ff4 2a5a23a
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 13:54:58 2015 -0500

    merge conflict

commit 6a93ff45d27458df206a0dea2cb3cbfe64a61820
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 20 13:53:42 2015 -0500

    updated presnetation, file tests

commit 2a5a23a7d8781a1aced8576bfac4da22477cfae0
Author: ms4985 <ms4985@columbia.edu>
Date:   Sun Dec 20 02:18:58 2015 -0500

    added more tests

commit 0d603b1e8406269ede1fc9a6ea4d6b714a4fa53d
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sat Dec 19 23:06:32 2015 -0500

    file IO works

commit 2557e441e0b3a4cc0c92b1d6a08ca9cadfb19c8a
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Sat Dec 19 21:26:09 2015 -0500

    added _ntests/arith3.stch, checks that you cant add chars to
```

```
    ints. Added _ntests/func2.stch, funcs w/o return type cause
    error.

commit 0cd0268a41ddbcc67a8f5bb21ffc17ef5f168fc3
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sat Dec 19 19:14:51 2015 -0500

    updated gitignore

commit e2b2651405022319038790b7ad8ef7e4fa22db72
Merge: 58aa1b4 443c22f
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sat Dec 19 19:13:18 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
    _lang

commit 58aa1b46b935835ab04af99604a6dc27c602fe35
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sat Dec 19 19:13:15 2015 -0500

    added FILE type

commit 443c22fd1528c3b2999c3d0a55d94f12f474443d
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sat Dec 19 19:08:05 2015 -0500

    More things now get screened before the struct

commit aae955dc54a06202de85586105c8ec2938e24a8e
Merge: eff5c5e d82e06d
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sat Dec 19 18:43:23 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
    _lang

commit eff5c5ea368de4d4899219c3aa1b13a52c74bb02
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sat Dec 19 18:43:05 2015 -0500
```

```
    added 2nd image for presentation, added new if test

commit d82e06d6b78ffba3105b7e9d97e94d9b18eb5e48
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sat Dec 19 18:42:36 2015 -0500

    Local stitch variables should not be put inside the struct
   now

commit 97eb3628befb3c8785eaf214599c6554fec22d84
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sat Dec 19 18:14:55 2015 -0500

    added presentation files

commit 067fe06aebdbd31d572c7e9a9471a21dfe4b2c91
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Sat Dec 19 17:21:07 2015 -0500

    no more CONST and  NULLin our language, also added global
   variable negative test

commit 66b262cfb5e01506bc9a449168e25e623b254d75
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Sat Dec 19 16:39:59 2015 -0500

    break works, tested in _tests/break1.stch

commit 3736edbef0dc5ff3df06cf3931f17a9ee63056f0
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sat Dec 19 16:08:16 2015 -0500

    Closer to getting array passing

commit 8f9e8967ffe7a06afa88f83b89ca0d90453f793a
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sat Dec 19 15:50:30 2015 -0500

    fixed unmatched accumulator tokens

commit 83bd3d36ae7130ec9160026054692aea835c8c2f
```

```
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Sat Dec 19 15:49:03 2015 -0500

    added mod operator to ops1.stch, checked logical operators in
    ops2.stch

commit 32662a7565735324058e8c84c7b2ed6405d0d3df
Merge: a5c3d93 7280502
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sat Dec 19 15:31:00 2015 -0500

    Merging changes

commit a5c3d9353644b6884ab83be67ab67a370004ff81
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sat Dec 19 15:30:51 2015 -0500

    Arrays are passing through

commit 7280502f0656cc75d20a6793279e7ff3e722897b
Merge: 40f096c a51ed05
Author: ms4985 <ms4985@columbia.edu>
Date:   Sat Dec 19 15:24:34 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit 40f096ce70eae7a48b09365befa0659c2d428d65
Author: ms4985 <ms4985@columbia.edu>
Date:   Sat Dec 19 15:24:24 2015 -0500

    added negative print test

commit a51ed052e8f8ddc8c70f0c7523e9e0464d4727c5
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Sat Dec 19 15:21:23 2015 -0500

    added _ntests/float1.stch, fails with multiple decimals in
   floating points

commit 60e63537836e38c965420a8279f50bfa699b3f80
```

```
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sat Dec 19 14:22:24 2015 -0500

    Passing arrays into multithreaded apps working 50%

commit dc56ccfb9de14951ef1a494f96db0088c21d75bf
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Fri Dec 18 18:24:25 2015 -0500

    added accumulator types intap, intam floatap floatam

commit df57730b475a7471e135f494c4a9afd1cc3b8ec5
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Fri Dec 18 14:32:14 2015 -0500

    fixed a testing issue

commit 11079547757834808d3e19b46d78deb6ae0b7877
Merge: b738f24 9fcef18
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Fri Dec 18 13:44:47 2015 -0500

    fixed merge conflict in stitch3_out.txt

commit b738f24919d8724e864fa1f58955760b32668df7
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Fri Dec 18 13:43:47 2015 -0500

    potential merge conflict

commit 9fcef180124171ce21c9d27bf58f191538457163
Author: Tim Waterman <watermantium@gmail.com>
Date:   Fri Dec 18 11:09:03 2015 -0500

    Passing variables into stitch

commit 3d587d1264a84e7a586ce82829da6d95dcb2bad8
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Fri Dec 18 00:21:10 2015 -0500

    fixed ordering on stitch2func matching
```

```
commit 96864c17443d2f972286b0a3724af83ffcd965bd
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 17 23:48:53 2015 -0500

    really added tests

commit 95cfc40432d3caf3e740a466bf44427c9b54b5ae
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 17 23:46:55 2015 -0500

    nested stitch loops work; added stitch tests 3 - 6

commit 2900c7f1c23a728b60e616407df48721046b868a
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 17 23:17:24 2015 -0500

    Stitch test 2 output fixed

commit a3537de8df1f2357baa55e2d8157d733785d79bf
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 17 23:16:59 2015 -0500

    fixed invalid return, fixed non-block stitch issues

commit 1aa8fd9f41f3b8cfc60d4d0716404e6b3bfa80aa
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 17 23:10:00 2015 -0500

    Working on multiple stitch loops

commit 081dc99c0659dd4a6f652a32d0e57f5bc6918f95
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 17 23:09:40 2015 -0500

    updated stitch2_out

commit 24a32f26e74a09b81f22ad720380f362d98200a1
Merge: f7318f8 93e2ac7
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 17 23:03:58 2015 -0500
```

```
    merge fix

commit f7318f8d4fd5ef755a7d7e7c753c5a0a861ee2a2
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 17 23:02:52 2015 -0500

    updated tests

commit 93e2ac70401a32852dbf28a6c4aa45e2ba4508db
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 17 23:02:27 2015 -0500

    Added multiple threadpools

commit ec977e821cfb62b6fb0bd8eae89fc4b289637494
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 17 22:45:44 2015 -0500

    removed struct/access; cleaned up old code

commit 503f05282e3f4493873eccbb92a2277b35046d97
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 17 22:37:57 2015 -0500

    Fixed stitch loops with fnames

commit 847bb3dc7381561e48c479054c71257c41b266e3
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 17 22:34:10 2015 -0500

    stch func naming works

commit d12ff2fa167e3016acb45fa3d5cf6377fd27bab6
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 17 22:25:49 2015 -0500

    updating stch func gen

commit d4acce8abd070fe0795ca20a08ecea821828dab6
Author: Tim Waterman <watermantium@gmail.com>
```

```
Date:   Thu Dec 17 22:24:53 2015 -0500

    Reworked structs to access stitch variables

commit 5408089ef6734ef9c569f425aa10f316a1b54045
Author: Tim Waterman <watermantium@gmail.com>
Date:   Wed Dec 16 14:08:58 2015 -0700

    Its going through syntactically, need to get variables passed
    in from headers

commit 64e35c5bfd99f33d705cd78041d642907d736c9d
Author: Tim Waterman <watermantium@gmail.com>
Date:   Wed Dec 16 13:51:47 2015 -0700

    Changing names correctly sometimes. Still not general enough

commit 747815f55365df192317f9c8a91e4697001cf050
Author: Tim Waterman <watermantium@gmail.com>
Date:   Tue Dec 15 10:35:39 2015 -0500

    Adjusted the variables in the generated for loops to be
   general

commit fc59e0265f8298ab26ba60417a6db8edca7acb95
Author: Tim Waterman <watermantium@gmail.com>
Date:   Tue Dec 15 10:26:29 2015 -0500

    Added thread blocking. Im an idiot

commit 5faf89299826880b76d63d469365b98e1a370395
Author: Tim Waterman <watermantium@gmail.com>
Date:   Mon Dec 14 23:59:11 2015 -0500

    Done for tonight

commit 91111abcda95102d45379e3d55fce9ba2d9b3f89
Author: Tim Waterman <watermantium@gmail.com>
Date:   Mon Dec 14 23:50:07 2015 -0500

    Still working on accessing passed in vars
```

```
commit ed3d01241778a94fcc5fb2912f542c84030f7fce
Author: Tim Waterman <watermantium@gmail.com>
Date:   Mon Dec 14 23:29:43 2015 -0500

    Need to figure out how to change the names of the variables
    in the stitch loops statement list

commit a051d852d2ab208436e673a6ec66e1a2e8f2a3af
Author: Tim Waterman <watermantium@gmail.com>
Date:   Mon Dec 14 22:46:43 2015 -0500

    Started to get threaded stitch working. Threadcount is a
    little wonky

commit 4b5101dbf24f5099115cf3a9a34b610f20818e02
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 13 19:04:34 2015 -0500

    stitch body now turns into a function, still need to get the
    nameing down

commit d523668973ee4357d92582fe0aa1abf91f3f8991
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sat Dec 12 20:50:04 2015 -0500

    redid the way that stitch funcs are passed, still need to
    generate the functions in the c_gen

commit 06ff138f3564f568a93df0d64a531eabb7542268
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Sat Dec 12 14:20:03 2015 -0500

    Update README.md

commit d45dd0e7e39b0075f69b07dfac6e04547537ad11
Author: Tim Waterman <watermantium@gmail.com>
Date:   Fri Dec 11 23:44:27 2015 -0500

    Started stitch->for code generation
```

```
commit 77bd9795055f04e9afc69cbd2b7f99577716da07
Merge: 05e6fbb bce1cd4
Author: Tim Waterman <watermantium@gmail.com>
Date:   Fri Dec 11 21:43:51 2015 -0500

    Merging with dans changes

commit 05e6fbb5b5b9300a1abcd9b06e7100b22b1f83be
Author: Tim Waterman <watermantium@gmail.com>
Date:   Fri Dec 11 21:43:48 2015 -0500

    Started work on for loop generation

commit bce1cd4399d50deff099affeee6f48fba0719002
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Fri Dec 11 21:21:47 2015 -0500

    little to no progress on the stch funcs

commit 4ed3f45d1f620971673d20dcfab4ac7545eece87
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Fri Dec 11 20:20:06 2015 -0500

    still working on stitch funcs

commit a23221534f8b6f860e571e88a4793dff2df32f70
Merge: 1ed3d07 1dff7ec
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Fri Dec 11 17:23:56 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
    _lang

commit 1ed3d07b8e35de80116f300f5d6f2c9fb57c3571
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Fri Dec 11 17:23:53 2015 -0500

    added stch_funcs

commit 1dff7ec6565c32c511260eb196435f15968e7d29
Author: Tim Waterman <watermantium@gmail.com>
```

```
Date:   Fri Dec 11 11:53:23 2015 -0500

    Finished matrix init. Need to remove debug statements later

commit 9c032762654796bff9347609ac1f218cb6ad8f47
Merge: a905ad3 952e6d6
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 10 19:47:33 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
    -lang

commit a905ad3de0802fe63588b511849e7475713dfa96
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 10 19:47:28 2015 -0500

    updated some tests, fixed list rev issue

commit 952e6d66e6601f9263c9cc39d1a95c190cca605d
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 10 19:46:54 2015 -0500

    Matrix init working 90%. Working on typechecking

commit 7163ee62b65f8b609cb7ad5face70e4bc61a5d38
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Thu Dec 10 19:43:00 2015 -0500

    added negate2 and negate3.stch

commit 148ca6dc15be4eb4eae58d011e4c78d2e4105729
Author: ms4985 <ms4985@columbia.edu>
Date:   Thu Dec 10 19:41:23 2015 -0500

    fixed rec1 test

commit 3e1db6e827105d913f01ef45877adc34db0f83f8
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 10 19:24:40 2015 -0500

    removed microc binary
```

```
commit ab1a394db56d434412ecd96343a0c8a30b2b4564
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 10 19:23:47 2015 -0500

    fixed snafu

commit 744381b513e3ba3fa008c3e99bb76901fed54b68
Author: ms4985 <ms4985@columbia.edu>
Date:   Thu Dec 10 19:16:44 2015 -0500

    added stitch and rec tests

commit a052841944b13bebaa4b6cd148d9efc5cbd409f8
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Thu Dec 10 19:16:01 2015 -0500

    added more tests

commit 4c9136b74b462cffa92ad6245ae75e5c0e9d20ca
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 10 19:12:56 2015 -0500

    working on stitch loops

commit f3cca0c68daf4fe582005673f22ca4c393c45cd4
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 10 18:42:04 2015 -0500

    updated stitch test

commit bb69889b8a08cb305318f24ab411627cb6bdc818
Merge: 4fc0380 9d063bc
Author: ms4985 <ms4985@columbia.edu>
Date:   Thu Dec 10 18:39:12 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
    _lang

commit 4fc038006a974131c958da7ac365a53c4e54aa7c
Author: ms4985 <ms4985@columbia.edu>
```

```
Date:   Thu Dec 10 18:39:00 2015 -0500

    added stitch loop test

commit 9d063bc7cbea95c3489fd2e65f78f4fb26105509
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 10 18:38:29 2015 -0500

    stitch loops redone in cast/analyzer/generator

commit e42e03b0b7c0c8e89400d1f2ecbd35d9b415da56
Merge: 4a3e470 20b60c4
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 10 16:50:20 2015 -0500

    "Merging with the syntax changes"

commit 4a3e4705a2f8cb2eb032cb79fe769ce545e03ebf
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 10 16:49:59 2015 -0500

    Added matrix checking. Now only init is missing

commit 20b60c4d0df664954390b5fda8bb422fada8255e
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 10 14:18:26 2015 -0500

    fix minor formatting

commit a449e53de6a31c9e6cf93e14dc00732152145260
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 10 14:11:11 2015 -0500

    fix conflict

commit a0e0636fcffbb966df1f9ef0f8019a9e97da48a7
Merge: f85e09e 209b7ed
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 10 14:08:03 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
```

```
    _lang

commit f85e09eeeb95251cfda76330ac68dd55a716df6a
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 10 14:07:13 2015 -0500

    bit of code cleanup in the sem-an, make it a bit more
    readable

commit 209b7edfb790ec5b8e94cfe3e40b6c121f2e8906
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 10 13:47:37 2015 -0500

    Array assignment in. 1D arrays should work 100% now

commit 121ba6810a11910ca7f1ca581c9f45773ad98edb
Merge: d55cb3b 0948ea2
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 10 13:16:10 2015 -0500

    Fixed merging issues

commit d55cb3b258de506e980ff64067fa72bb39726d1e
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 10 13:12:07 2015 -0500

    Fixing matching issue with arrays; error messages not
    accurate

commit 0948ea2b5805d2856e43f584aa6df84dd89d124c
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Wed Dec 9 18:09:11 2015 -0500

    questions about matching

commit 10cd1cb8fba3f2dfcbed3e071a71e9d21d096bf6
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Wed Dec 9 17:17:09 2015 -0500

    test suite now echos total number of tests and how many
    passed
```

```
commit 68326e9c504acff661b2ad5bb384d5b3bc4b2d2e
Merge: cd26dcf ae9e8b6
Author: Tim Waterman <watermantium@gmail.com>
Date:   Wed Dec 9 16:21:40 2015 -0500

    Pushing the negative array tests after merge

commit cd26dcf1753f24f1ba0ce9216aee5f0bc43bf0ce
Author: Tim Waterman <watermantium@gmail.com>
Date:   Wed Dec 9 16:21:30 2015 -0500

    Pushing array negative tests

commit ae9e8b6eba22a76202dfcc73b03e546af22ad511
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Dec 8 18:47:54 2015 -0500

    minor commenting

commit 28ec92af50354d748234b76847a993123e5a9e6a
Author: Tim Waterman <watermantium@gmail.com>
Date:   Tue Dec 8 16:35:29 2015 -0500

    Added size checking for array init

commit a88a2a4d696ee1ec8464ab4bec5a80056c8a451a
Author: Tim Waterman <watermantium@gmail.com>
Date:   Tue Dec 8 15:55:32 2015 -0500

    Array initialization working

commit b116aa4bd0ee75feec5a11df97bbe1ddd93c0112
Merge: e052491 aadd3b9
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Dec 8 14:14:47 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
    _lang

commit e052491cdb41118329d8f028314fb300d3c2c5de
```

```
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Dec 8 14:14:40 2015 -0500

    added exit() + tests; added commenting tests

commit aadd3b987c848a1fc97d17524376ff6d3519bed6
Author: Tim Waterman <watermantium@gmail.com>
Date:   Tue Dec 8 14:05:32 2015 -0500

    Added array indexing to the print statement

commit 17e1a695296b3dd1cb19110b1a1d4a81678a3f64
Author: Tim Waterman <watermantium@gmail.com>
Date:   Tue Dec 8 13:57:00 2015 -0500

    Added in array indexing expressions. Need to add them to
   print

commit 93272f2330e441ac28906db16dab3067503dacd6
Author: Tim Waterman <watermantium@gmail.com>
Date:   Mon Dec 7 23:04:52 2015 -0500

    Added more array items. Working on 1D array init

commit 05ca28f77603d036ed9b4adf9023c473926e9fff
Author: Tim Waterman <watermantium@gmail.com>
Date:   Mon Dec 7 21:18:18 2015 -0500

    Added two dimension array decls, working all the way through

commit 39a8a4d403c7083a6049569b97a7f7ae9ae4dd88
Merge: 73bc7ff 3527842
Author: Tim Waterman <watermantium@gmail.com>
Date:   Mon Dec 7 19:06:26 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit 73bc7ff3374bf6492fd0cc24ce4f342e03ac1914
Author: Tim Waterman <watermantium@gmail.com>
Date:   Mon Dec 7 19:06:15 2015 -0500
```

```
    One dimensional array declarations working, parse->print

commit 3527842aa87c50070f6b76ddafc6768e548479e7
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Mon Dec 7 17:17:26 2015 -0500

    file cleanup for negative tests

commit 9b083ee0ff021608921800c4a355de2f901db6dc
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Mon Dec 7 17:17:12 2015 -0500

    updated negative tests

commit 6470897bd762270847c1f1768906851d6d57b5f8
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Mon Dec 7 17:03:00 2015 -0500

    test suite works with negative testing (goes in _ntests);
   updated headers

commit 8f0482eb4b19acd7a8a4c8e6e758a2541703a7b4
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Mon Dec 7 16:24:01 2015 -0500

    Update README.md

commit 5ca219aa8fa3b010b76f13087198ee7baf54a2ae
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 6 17:28:30 2015 -0500

    error() added Stitch.  Added hello2 test to confirm this

commit cf3e7898576149a1d37a52f30a36cc898e13a460
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 6 17:08:57 2015 -0500

    print works

commit bbd481ecf6e544fb3d6796b30f75ef1c6dc22f93
```

```
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Dec 6 16:35:05 2015 -0500

    print works for int, float, char, string, id; does not work
   for other expr

commit 6f52d3db9df12d425aeb0a7686579766afbde676
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sat Dec 5 17:37:09 2015 -0500

    print works for ints now, updated all tests accordingly

commit 8049b3dc74d171068c6b6c79cafaf272c7013caf
Author: Tim Waterman <watermantium@gmail.com>
Date:   Fri Dec 4 22:02:46 2015 -0500

    Quick fix for if statements. Need to rethink how we print

commit d8cb215629af8f4eb180de00cfb578b91f9ebd0d
Author: Tim Waterman <watermantium@gmail.com>
Date:   Fri Dec 4 21:41:55 2015 -0500

    Fixed the stitch compiler chain, started adding print

commit 25791cdb3d156e9bc3e21b729222013338243af5
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Fri Dec 4 13:49:11 2015 -0500

    working on c_call - find_func_sig working (?)

commit d514d8c32ea80308150b8391a8c3d9d1a2be49c0
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Fri Dec 4 13:41:01 2015 -0500

    working on c_call - find_func_sig

commit 8dfea1a4056bce6cfa0077746ce7a0cd9cc048ae
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Fri Dec 4 13:34:03 2015 -0500

    working on c_call - need to finish func args
```

```
commit c6afbb284ad07a4782191f0ee2d6a57776e7a0ce
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Fri Dec 4 11:47:27 2015 -0500

    added better error for check_assign2, tweeked cleanup of test
    suite

commit 01a92d5332fd6be8b381c92124d059797149c379
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Fri Dec 4 01:40:21 2015 -0500

    fixed assign

commit 0d18e5f2dc10a4b42bc9be3a04bbd306cbaeee42
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 3 21:27:09 2015 -0500

    Debugging variable issues

commit b41f84754f9e2e01e7ccf665bf95488a7692ae30
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 3 20:51:12 2015 -0500

    For loops correctly accept expr_opt

commit c07013b4c847fdc45bca258c705c3fb517d1c495
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 3 20:49:53 2015 -0500

    Added testing file

commit 2cdd1126be26fd365d05714654f86f9c8b04ad47
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 3 20:38:05 2015 -0500

    Generator issues, but everything seems to compile

commit 308b52d1d7fe1fd1c8c85d53786d07ff0b151928
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 3 20:27:00 2015 -0500
```

```
    fixed gen

commit 479c8701c4df07f1f37653b9d5013ef21b2c2fa5
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 3 20:15:58 2015 -0500

    Started the C code generation file

commit 9a6bcf3ecd6ea00a525d4dd2bbff7411896e1a16
Merge: e6246b0 95fde55
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 3 19:53:26 2015 -0500

    Fixed merge bullshit

commit e6246b0ed9654c03d8617998a22b81f6b9f97877
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 3 19:51:26 2015 -0500

    Stitch thing working

commit 95fde554d5d23170047438a8f47c2abb8340db65
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 3 19:49:55 2015 -0500

    sm-an works again

commit cdb0b1ac080b6b897e0c89f335f18195c543576b
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 3 19:36:34 2015 -0500

    Makefile and compiler edit

commit 7f15f7f49eef9ef9d18ae1eb0e82ea2fdad05a7d
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Thu Dec 3 19:35:03 2015 -0500

    added check_for

commit 61f94774363231958a4dd178850459c9a0348f27
```

```
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 3 19:31:19 2015 -0500

    working on program

commit 93ea0d82be4bb0d166e65b30b108a98b91633070
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 3 18:39:29 2015 -0500

    making progress

commit ebfa2ec2e5af5e9ad9956b2fa564fb6dd5d3d508
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 3 18:21:14 2015 -0500

    w

commit 67e40f8420c6d7f8df9cff3f0763d905713d371d
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 3 17:49:37 2015 -0500

    w

commit a07af01a4af200d1773db6ba89bad67b1eae974c
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 3 17:40:53 2015 -0500

    fixed

commit 86db74aff9b3f8ce04b91284c85e3150ad368362
Merge: d7f53b2 07a24c3
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 3 17:25:06 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit d7f53b2e53407b5cea1a5ddbf94e457d7ad3bea5
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Dec 3 17:24:58 2015 -0500
```

```
    working on sm

commit 07a24c353a57e3e50d5cc91e356faa2408a2bdc2
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 3 16:49:28 2015 -0500

    Checked if expressions, while semantic checking done

commit 09a146ab66b698bcedab996f2481847d5beeb315
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Dec 3 16:33:34 2015 -0500

    If semantic checking checks for int type expr

commit a367ac78d4acbd95110ad958d4fa8d41180055cf
Author: Tim Waterman <watermantium@gmail.com>
Date:   Wed Dec 2 20:05:25 2015 -0500

    Semantic checker compiling with the pieces we have, a lot has
    been commented out

commit c68b1dfa5fb2bf8e469f1c6f6b0ec972aeddb2f3
Author: Tim Waterman <watermantium@gmail.com>
Date:   Wed Dec 2 19:11:00 2015 -0500

    Fixed optional error

commit 19d4e043bfae7acee69c090495e91220219bc6bf
Merge: af3a9ec ca26c84
Author: Tim Waterman <watermantium@gmail.com>
Date:   Wed Dec 2 17:57:32 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit af3a9ec4dbb6806eba0b31d7063d677efa905844
Author: Tim Waterman <watermantium@gmail.com>
Date:   Wed Dec 2 17:57:23 2015 -0500

    Fixed compile stuff
```

```
commit ca26c84fd8e274c1db816d3db9330c39a52b473c
Merge: f8be026 83f1175
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Wed Dec 2 17:48:48 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
    _lang

commit f8be0264b1d229d54bd13b82deb87814179af5c7
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Wed Dec 2 17:48:40 2015 -0500

    updated makefile

commit 83f1175e536b841796b971324361aa78659fc6ac
Merge: 53282cb 153fff7
Author: ms4985 <ms4985@columbia.edu>
Date:   Wed Dec 2 16:13:04 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
    _lang

commit 53282cb6ab31b0a1ae299be89a8c8b160f8fe299
Author: ms4985 <ms4985@columbia.edu>
Date:   Wed Dec 2 16:12:58 2015 -0500

    fixed up sem tests

commit 153fff71062d4532dd8d2dade9aad2d58401fc34
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Wed Dec 2 16:10:39 2015 -0500

    updated comments

commit 3acc1a9c8200480e57791b250e92c44de5a0e42a
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Wed Dec 2 15:02:22 2015 -0500

    commented sem_an, still need to do check_for, fix
    check_var_decl
```

```
commit 50f444b4ee818db79e5af2f86d10c33fc219b1a2
Author: Tim Waterman <watermantium@gmail.com>
Date:   Wed Dec 2 14:37:09 2015 -0500

    Added back access operator and struct keyword

commit 24723bb66ce7c484bbd1d3eec2c582d1e76ff049
Author: Tim Waterman <watermantium@gmail.com>
Date:   Tue Dec 1 23:39:09 2015 -0500

    Arrays now can assign individual elements

commit 87bdbbe5804eb72d1c4d4f0a6cb00dbf21e9f92c
Author: Tim Waterman <watermantium@gmail.com>
Date:   Tue Dec 1 23:10:06 2015 -0500

    Initial array declaration passing parser tests

commit a6f399113c32bac2915678d66ea64a3f5d878973
Author: Tim Waterman <watermantium@gmail.com>
Date:   Tue Dec 1 12:55:27 2015 -0500

    Fixed pretty printing of chars from ast

commit 53166bf86b82e5994464cfc71b9ff1a69e52c094
Author: ms4985 <ms4985@columbia.edu>
Date:   Tue Dec 1 11:05:06 2015 -0500

    added the actual semantic checks to test

commit b44f18dd7fa5e2c32d5fede5298f184de76f9ab4
Author: ms4985 <ms4985@columbia.edu>
Date:   Tue Dec 1 10:59:14 2015 -0500

    added simple semantic analysis tests

commit 07a81ef1ecfb6d069966a363d0dbcba5dd35b9c4
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Mon Nov 30 12:11:38 2015 -0500

    semantic analizer compiles, still need gen code
```

```
commit 5e1fba16104839ac3964bbf7cb73198416e61192
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Mon Nov 30 12:04:15 2015 -0500

    finished sm-an

commit 2aefc5892f392b9d6833fbd1932ac183f03e6423
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Nov 29 17:08:26 2015 -0500

    need to still add init_env and chck_program

commit c1d8dff46a438722f3d181b29451d32b899a7713
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Nov 29 16:46:31 2015 -0500

    working on sem_an

commit 3df7000bffe087e4d01d21b3b7dd4dd19ed00965
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Nov 24 13:20:22 2015 -0500

    working on sem-an

commit 135caf8726134865d43f8ed6a65ebdcad98b313e
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Nov 24 13:05:15 2015 -0500

    started on semantic analysis

commit bd34b67afac6cf13c42b59d984e32bb1deab2bd4
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Nov 22 20:21:05 2015 -0500

    created cast, added type dataType refactored the ast for
   consistancy

commit 79b9ebc687898a82a6f012cd69d9e65632251bec
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Nov 22 19:21:01 2015 -0500
```

```
    started on env and complier

commit 2eb53e0ac96e5fc98d4a7728c7247578b730b7ad
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Nov 19 18:06:00 2015 -0500

    updated .gitignore, added makefile for stch_headers library

commit 34301c36c591a75e5ee0ae8c7682d13091966a9d
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Nov 19 18:04:48 2015 -0500

    Added new loop tests

commit 7bb3f01fb70b60d35bb1d748605e20737757b1cf
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Nov 19 17:57:16 2015 -0500

    added function test 6 - check for empty return

commit b491f3d154bda7dbf9d91620a46491ee8c12f9e3
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Nov 19 17:56:10 2015 -0500

    fixed some stuff

commit 8171503e3921b3e15907bf06770c9bf47f0ad811
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Nov 19 13:41:46 2015 -0500

    removed gen code diffing from test suite

commit 2fbbb1d5a434528ff40a6205f50ef1d616a53901
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Thu Nov 19 13:14:49 2015 -0500

    Update README.md

commit bf76fbf0e0040e5ae8e3dbb170d3b2e3f9a8a9d8
Author: Tim Waterman <watermantium@gmail.com>
```

```
Date:   Tue Nov 17 21:20:24 2015 -0500

    Fixed shift/reduce error caused by stitch loop. All tests
   still work

commit c2892cf56d57fe0aca1aaaa17a69766431094100
Author: Rashedul Haydar <rhaydar8@gmail.com>
Date:   Tue Nov 17 15:46:25 2015 -0500

    updated meeting notes

commit 33fa100ab44678da625a69ea7c9acb64cfefdc4d
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Tue Nov 17 15:07:03 2015 -0500

    Update meetingNotes.md

commit 4c46e088822371df43782b93cd61bacf62a02c5a
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Nov 17 12:01:17 2015 -0500

    added two new tests

commit b5d767afb03ba0b1d3e058b49e528a1f9d7b7aab
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Nov 15 21:40:38 2015 -0500

    added fucn1 targets

commit efc6214ac58989d432b07dc0b42a950321ad25bf
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Nov 15 21:35:15 2015 -0500

    fixed newline issue, removed .dSYM

commit 876191de3d0b7a13324b69c837228ac13289cde0
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Nov 15 21:30:47 2015 -0500

    AST changes to allow vdecls
```

```
commit 6edbc79588c29679de29a66d52a2ea9ae9ceb71d
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Nov 15 21:29:58 2015 -0500

    updated gitignore

commit a656e75075fea22e29fbc0855eb76f9caa6ce17f
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Nov 15 21:25:02 2015 -0500

    tester now recompiles everything

commit 856235421ca473adb5e661a6ffbea7565b862aa7
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Nov 15 21:15:15 2015 -0500

    added new tests (spoiler, we pass)

commit 7cfbd96c49b9a5b7d7e06e45f40cbbf2123ee99e
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Nov 15 20:48:02 2015 -0500

    Changed ast to use printf

commit 1740d751c0d627fb4018e5887b1d7d7f3eb1849a
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Nov 15 20:45:22 2015 -0500

    added dummy files for _log and _bin

commit 892ca104b7aacf09a887b6d892666e9e0cf1d0cb
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Nov 15 20:27:58 2015 -0500

    fixed int x = 3 error

commit 27f59871b4707649cb1b6deee7c9ccc421c781b7
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sat Nov 14 00:07:56 2015 -0500

    test suite now works, also folded in color, etc. from parse
```

```
    tester

commit e9cf2179c20fafba171fe9cc4389213cf41c2370
Author: Tim Waterman <watermantium@gmail.com>
Date:   Fri Nov 13 12:16:20 2015 -0500

    Fixed tester cases, shift reduce still happening

commit 7fb2182fa4c9569011cfade64c772468633cb984
Author: Tim Waterman <watermantium@gmail.com>
Date:   Fri Nov 13 00:19:56 2015 -0500

    Made ptest more extendable with functions

commit 000a738da56f1e36a2b1fa686672388d86370d9f
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Nov 12 23:42:58 2015 -0500

    Colorized output of ptest suite

commit 682a5728a1d43a323c2900ecbfc990ba34edccd6
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Nov 12 22:50:04 2015 -0500

    fixed 8 SR errors

commit 490aa2fdb761d84878860fc2056e84f7d37ff937
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Nov 12 22:43:02 2015 -0500

    Ptests report what they are; while/stitch added

commit d24d0d3645b6fa931632aabd38e5b89473ed35d8
Author: ms4985 <ms4985@columbia.edu>
Date:   Thu Nov 12 22:18:06 2015 -0500

    added test doc

commit c6dee2286fb8c13976c8d4c7c342a762811a9b5d
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Nov 12 22:17:41 2015 -0500
```

```
    hello, world

commit ef6c05f24695a445c963fd69a6f9ca89848a4eb6
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Nov 12 22:02:01 2015 -0500

    Parser test script updated

commit 71d238926bffe47ecf9d150e01ac641de54fad5c
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Nov 12 21:55:39 2015 -0500

    Stitch includes headers

commit f4bf3f3c98f8526dbb90b8c8a924059df688f941
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Nov 12 21:48:51 2015 -0500

    functions working

commit a9a66840ce6149982bbfc426a2444bb3430fc506
Merge: 80823a2 a41a639
Author: ms4985 <ms4985@columbia.edu>
Date:   Thu Nov 12 21:45:24 2015 -0500

    Merge branch 'master' of https://github.com/danhcole/4115
    _lang

commit 80823a268592ff0e7ef847970d3fd349e7f9eca6
Author: ms4985 <ms4985@columbia.edu>
Date:   Thu Nov 12 21:45:10 2015 -0500

    added more tests

commit a41a639f64cd7ce4928e429964cdd38bc2875bd2
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Nov 12 21:28:45 2015 -0500

    minor fix
```

```
commit 2788f4c6905199da5809b05604683aead1f52021
Author: ms4985 <ms4985@columbia.edu>
Date:   Thu Nov 12 21:28:13 2015 -0500

    added test suite for parser

commit a7c5865238d584ddfa294f3e74a949e71bae8c52
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Nov 12 20:48:20 2015 -0500

    working on functions

commit 7e594a1be51209e15a2036e42d675c57abeaadcd
Merge: 337987d 453f44c
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Nov 12 20:47:42 2015 -0500

    Fixed stitch.ml issue

commit 337987de8088fb08547c57def1e1a38a4445acec
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Nov 12 20:45:13 2015 -0500

    Stitch.ml taking in filename

commit 453f44c460ce2284d3bbb6746969955484a8de44
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Nov 12 19:30:40 2015 -0500

    unordered vdecl statments works

commit c611d232489da46e7980218d78a36d28d3a82786
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Nov 12 19:05:52 2015 -0500

    restructured vdelcs

commit d7a3b80647b4b5ee1f8422d2df0c0619842556bd
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Nov 12 18:14:18 2015 -0500
```

```
    added stitch to pretty print

commit 372687d8979d718aa95c77d80043ca6167bc15dc
Author: Rashedul Haydar <rhaydar8@gmail.com>
Date:   Tue Nov 10 19:09:15 2015 -0500

    fixed formatting

commit ed6e4eb55757b017f2314121e009a6b463e05127
Author: Rashedul Haydar <rhaydar8@gmail.com>
Date:   Tue Nov 10 19:08:04 2015 -0500

    added notes from 11/10 meeting

commit cd71c4551c3dcca992c8a558a43e3fc41aa24e57
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Nov 10 16:15:16 2015 -0500

    compiler compiles (with nothing in it_
    )

commit 6d743a00b1cbf24b92b34eaddef8c9cd6d69a2c3
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Nov 10 15:37:34 2015 -0500

    2 more errors fixed

commit 6dba4d67453ecb49007b733ad32ae9412fe3698e
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Nov 10 15:21:01 2015 -0500

    updated

commit 31538c38a2cd5f4537ff694a536f0f4173f4d0eb
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Nov 10 13:44:33 2015 -0500

    fixed some issues

commit 68c76a75ffef4e3ba3ac323f3b44c2b23a38c76e
Author: Daniel Cole <takeitfromthedan@gmail.com>
```

```
Date:   Tue Nov 10 12:28:33 2015 -0500

    added Makefile

commit d4b388634339bf16d57a8819d4e803545fb5db20
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Nov 10 11:43:35 2015 -0500

    test update 2

commit 87a91772064c8e9c6b0ebf32bd65ddd22ee14b44
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Nov 10 11:42:21 2015 -0500

    test update 1

commit 7adba58bdd0c1a33c393b8b8636759ab57e7d0f1
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Nov 10 01:47:55 2015 -0500

    updated test script

commit a26d8ff170dbf9f72e9930afe0920464f4505be4
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Nov 10 01:42:58 2015 -0500

    test suite added

commit 23eb1db66e5506a86aaebc06a0b94b2b909d2b18
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Nov 10 01:40:37 2015 -0500

    tests

commit 450976a82bf9a87cb4686f4827f8fc4d6197d8aa
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Mon Nov 9 18:06:59 2015 -0500

    Delete hello1_target.c

commit 85e07a535c066c960b7da02da713c58b11574e25
```

```
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Mon Nov 9 18:06:45 2015 -0500

    Delete stch_headers.h

commit 3615729f345a3aa9049607c163c6021dc038740b
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Mon Nov 9 18:05:53 2015 -0500

    started work on testing

commit 736cb3ac8ed69d787e37299afdbdafce2eec822e
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Mon Nov 9 17:07:46 2015 -0500

    hw test target

commit 78e30de0b28a2dec1a323eee367b006436cf50b4
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Mon Nov 9 15:52:21 2015 -0500

    Update stch_compiler.ml

commit 387cf5404148befd486af3268fd9ce394376c4c5
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Mon Nov 9 15:22:54 2015 -0500

    Create stch_compiler.ml

commit 997ff10affded68449ea05e3cedbeb290ea17acc
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Mon Nov 9 15:21:47 2015 -0500

    Create hello2.stch

commit ac2ab8a9a9cf0b094acf5e897812d75fefdd6d30
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Mon Nov 9 15:16:28 2015 -0500

    Create hello1.stch
```

```
commit 3baf3d29959ee88b8bb4afa3542c351947e17965
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Mon Nov 9 10:54:22 2015 -0500

    stch c headers and func defs added

commit c86292b09b0ddc5ba33a71cd86471c4a318ecfa2
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Sun Nov 8 20:06:36 2015 -0500

    Update stch_headers.h

commit d23f256f266c4573c5f9bc83bc4ad0416b76c20f
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Sun Nov 8 20:01:46 2015 -0500

    runtime questions for 4/10

commit 6b47151d175862256a05adf28b96b690a706a13e
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Thu Nov 5 17:47:35 2015 -0500

    Update stch_headers.h

commit 47eea775ef5cd9719b20b540683f9cfffe16173b
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Thu Nov 5 13:16:49 2015 -0500

    Started work on header

    This header file will be auto-included at the beginning of
   any c file generated by the Stitch compiler.

    It should include all includes, defines, and structs needed
   by every Stitch-C program.  Program specific functions will be
    generated separately.

commit bad6110bf902b80c6d6dab96e9361b3a422edd31
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Thu Nov 5 12:13:16 2015 -0500
```

```
    removed bitwise, unary operators

commit a7ae8c95120edee5c1a3fec12c7ffb9d57ea9606
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Thu Nov 5 12:12:01 2015 -0500

    removed bitwise, unary operators

commit 2568ef7fd2a282c838dc9605768a21a04510d1f1
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Thu Nov 5 12:08:32 2015 -0500

    removed bitwise, unary operators

commit 3febda5584311329ff4ce362a669b3e513a34cd5
Author: Rashedul Haydar <rhaydar8@gmail.com>
Date:   Wed Nov 4 21:43:15 2015 -0500

    added my notes from meeting

commit 279aade3ad61b3cb653b32e5b98b92e5ca6f2058
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Tue Nov 3 15:23:42 2015 -0500

    Update meetingNotes.md

commit 29e5576bae82533f51bcb7939d5e901f503beeb8
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Sun Nov 1 15:39:23 2015 -0500

    arrays work without any shift/reduce errors

commit 30551c08218c896be4888779dec8455f7e716122
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Sun Nov 1 14:51:48 2015 -0500

    fixed typos in arrays

commit c1f2800dd4f252149b00288c921c8001775dbcd7
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Sun Nov 1 14:46:50 2015 -0500
```

```
    started to add arrays

commit 10725757d0c9b2e95172eac8f86fbd58b6c0ef52
Author: ms4985 <ms4985@columbia.edu>
Date:   Thu Oct 29 19:36:02 2015 -0400

    took out Call from ast

commit 54c7cd249affffb534054d1115ad9569c7eddaa8
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Thu Oct 29 19:10:10 2015 -0400

    fixed parser, no more shift/reduce errors. need to add arrays
    still.

commit 633eb9c0ea5b060e4dd6e1c724148f39d811370b
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Oct 29 18:50:49 2015 -0400

    updated typenames, updated gitignore, added singer

commit ff2e47bf4b1be678470743b896eafacff701d0a8
Merge: f04092d 53193cf
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Thu Oct 29 18:43:20 2015 -0400

    Merge branch 'master' of http://github.com/danhcole/4115_lang

commit f04092d2d0feb89c8ecb6fe910ba5b3d51d70eb2
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Thu Oct 29 18:42:47 2015 -0400

    got rid of ftype

commit 53193cf3451f67dedc5d2fa74150ddc9dd73f433
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Thu Oct 29 12:52:43 2015 -0400

    Create testDoc.md
```

```
commit 700b222ad81847077ae826d37ae73fa846c97584
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Thu Oct 29 12:27:32 2015 -0400

    updated meeting times, added link to LRM

commit afaada4a0582941f3997cd30c996616e7515f81a
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Oct 29 12:24:59 2015 -0400

    fixed gitignore, started basic compiler toolchain

commit ef0afdbc41a3fac03b12d60fdfc00721247efb97
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Wed Oct 28 18:09:16 2015 -0400

    added .gitignroe, trying to keep things clean

commit 79b78ad499571f143162585822d321307965a471
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Tue Oct 27 17:11:03 2015 -0400

    Update meetingNotes.md

commit 3c0d24d7c9a0be915a8ec0c0a1bd4f6f5702a959
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Oct 27 14:14:01 2015 -0400

    fixed dotproduct

commit b969752af73c8f4f775615c75be026381f13eb2c
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Oct 27 13:03:53 2015 -0400

    dotproduct updated

commit 311db285615ede90022bd1da46c685d6c74510d7
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Tue Oct 27 11:57:14 2015 -0400

    removed .* operator
```

```
commit 0f4f13a52d9a05594e2403db0f54f5deada242e7
Merge: 6ee4639 313a484
Author: Tim Waterman <watermantium@gmail.com>
Date:   Mon Oct 26 14:03:45 2015 -0400

    Merge branch 'master' of https://github.com/danhcole/4115
    _lang

commit 6ee4639e42850fdd4d23f866123e48f0c4c227df
Author: Tim Waterman <watermantium@gmail.com>
Date:   Mon Oct 26 14:03:01 2015 -0400

    Fixed myAdd, working on matrix mult

commit 313a4846a39797db7f10fb758665416f029c29c8
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Sun Oct 25 16:57:29 2015 -0400

    most of the parser is done, have 4 shift/reduce errors

commit cec57449b85e151ac636ce3d91cc114b1290e2b0
Author: ms4985 <ms4985@columbia.edu>
Date:   Sun Oct 25 16:48:54 2015 -0400

    backtracked

commit 29e73d01c4d40260d8799672a57ea411d15a8c9a
Author: ms4985 <ms4985@columbia.edu>
Date:   Sun Oct 25 16:46:47 2015 -0400

    assign now takes a vdecl

commit 3ac86e93cbb8bbf670cebbc8f20add795b066151
Author: ms4985 <ms4985@columbia.edu>
Date:   Sun Oct 25 16:36:15 2015 -0400

    added a vdecl struct to account for multiple typenames

commit f8932cf5aa574b393f11f22a8784f028ddd86b01
Author: ms4985 <ms4985@columbia.edu>
```

```
Date:   Sun Oct 25 16:00:05 2015 -0400

    added stitch stmt

commit 24be4c050448c6e1c3149884483d1457bc44f6e5
Author: ms4985 <ms4985@columbia.edu>
Date:   Sun Oct 25 15:38:44 2015 -0400

    minor fix

commit 02441f86411db60084a486b1f8aa22a40a93ef90
Author: ms4985 <ms4985@columbia.edu>
Date:   Sun Oct 25 15:37:09 2015 -0400

    minor fix

commit ddfcd2f74fb474922dd7fbb90dde18a55f037da4
Author: ms4985 <ms4985@columbia.edu>
Date:   Sun Oct 25 15:36:33 2015 -0400

    fixed unary types

commit db2373159a068a8f3175cd8e16127de19a4c2732
Author: ms4985 <ms4985@columbia.edu>
Date:   Sun Oct 25 15:29:31 2015 -0400

    added data types and fixed fn names

commit ce4ebcb7a3c699e7a4396707dc4b7609890920e2
Author: ms4985 <ms4985@columbia.edu>
Date:   Sun Oct 25 14:47:55 2015 -0400

    added stitch ops

commit e4778813c18a3a4dcfc6729cf6d3695519d9347f
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Sun Oct 25 14:19:03 2015 -0400

    added all associative operators

commit 353fa76c5504261f3eb526528e3e1173ebfbde7f
```

```
Author: Rashedul Haydar <rhaydar8@gmail.com>
Date:   Sun Oct 25 13:51:47 2015 -0400

    updated parser

commit 186cf5ccc8d56277996cc438e599d84a218ed3e9
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Oct 25 13:20:03 2015 -0400

    actually save the updated scanner first...

commit 06fecf11ab9e99ecc1a918dc07bc44975545211e
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Oct 25 13:19:13 2015 -0400

    updated scanner

commit 4cd70880397079e114f45268e920d021621c8643
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sat Oct 24 17:31:53 2015 -0400

    minor scanner fix

commit 25ba2f17e58d1733dc6dcd274d18276c557ed694
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Sat Oct 24 17:25:02 2015 -0400

    Update README.md

commit a5a2637934fa321c1cc7ead3489eaf6ff5daaca3
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sat Oct 24 13:28:02 2015 -0400

    Adding N to each element - finished

commit 0d507d027e00a643f846030c150c5673d3769769
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sat Oct 24 12:57:02 2015 -0400

    Pthread skeleton working
```

```
commit e906286b4ce487ee709793983089670832 54cfba
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sat Oct 24 12:21:46 2015 -0400

    updated scanner

commit 2c3592b23232c5ac1e6e369d98cace990ecf9aa7
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Fri Oct 23 19:50:00 2015 -0400

    Update dotproduct.stch

commit af945319fb88f0158beb1da35b902f7b23d7cb56
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Fri Oct 23 19:46:46 2015 -0400

    Update dotproduct.stch

commit 7d705960d1fe7c9c208f5969ab35a72dd6cf194c
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Fri Oct 23 19:45:12 2015 -0400

    Create dotproduct.stch

commit 09c1e8527116e18d64f6db1545948cdeaee90dc3
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Fri Oct 23 19:28:15 2015 -0400

    Update ideas.md

commit c83c68aec8a2565f0b5ada270bcacc21c87e0a50
Merge: edbe1d9 308e8b5
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Fri Oct 23 18:01:54 2015 -0400

    Merge branch 'master' of https://github.com/danhcole/4115
    _lang

commit edbe1d939afbb7c1a7350c7c5b4e70794d584a14
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Fri Oct 23 18:01:46 2015 -0400
```

```
    scanner mostly done, parser started

commit 308e8b521e95b896cc172b621e0ec776a3d5921a
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Thu Oct 22 18:17:55 2015 -0400

    Update ideas.md

commit 4e97fbf8cd985882cacfd187ac9cd89dadeca68f
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Thu Oct 22 12:48:32 2015 -0400

    Update ideas.md

commit 74380b1db70f1ac201a4defd9970df7b7fb7b5b5
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Thu Oct 22 12:36:40 2015 -0400

    Create ideas.md

commit ddf5d1e7fd533e11d0de8bf86cd7ff120fbf1f87
Author: Tim Waterman <watermantium@gmail.com>
Date:   Wed Oct 21 17:04:20 2015 -0400

    Added microc sample code

commit ef9f52ca748e31aedf7cb00b69c27f1566d96c0b
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Tue Oct 20 15:34:04 2015 -0400

    Update meetingNotes.md

commit c1a12e6d4897d9d6b115897ce809c1490f2e4187
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Sun Oct 18 16:04:56 2015 -0400

    Update TODO.md

commit 83422c57deb7b0f60b9ee1b6c70c104590a2bc36
Author: Daniel Cole <dhc2131@columbia.edu>
```

```
Date:   Sun Oct 18 16:03:38 2015 -0400

    Rename TODO to TODO.md

commit ffe0d8f1291fed06c4a2c13f45b0186d5ec956d9
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Sun Oct 18 16:03:21 2015 -0400

    10/20

commit 1d46f16bf292decf7f4a1a0175863557880a1fe3
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Sun Oct 18 15:55:04 2015 -0400

    questions for 10/20

commit 21154b8fc5419395982feb757cd854945c85f80a
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Oct 18 15:45:36 2015 -0400

    Examples changed

commit 57c0774b562cae68ce2f3ada374ac46fb4415c60
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Sun Oct 18 15:44:13 2015 -0400

    Update examples.stch

commit 49217d102ca937c72e7ace9e915c16a5affb2501
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Sun Oct 18 15:07:01 2015 -0400

    Create examples.stch

commit 117b5f74a9449bd7a7475330a508bb9d6278b69e
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Sun Oct 18 15:02:24 2015 -0400

    Update LoopExample.stch

commit 0931cc4540a6940cd9eb78e4b6a8ef497f190de1
```

```
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Fri Oct 16 00:16:23 2015 -0400

    Update LoopExample.stch

commit 9a681ac387d7e8bc22cb499d5c847537f76651f6
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Fri Oct 16 00:05:35 2015 -0400

    Update meetingNotes.md

commit 69795aec00ccef4cb233ae3763959a945bf7e682
Author: Tim Waterman <watermantium@gmail.com>
Date:   Thu Oct 15 22:48:25 2015 -0400

    Some early thoughts on syntax stuff

commit ccb5b7c49431e859085a857bd0ad6e7cc75f2e0d
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Thu Oct 15 17:23:26 2015 -0400

    Update meetingNotes.md

commit a4ddfb53c3ec166017acbc2046707e5c1d423f39
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Thu Oct 15 17:22:19 2015 -0400

    cleaned up notes, reorg.

commit 337e1fc431a6d35c40877fc4870a39ab63825a99
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Thu Oct 15 10:18:30 2015 -0400

    Update MeetingNotes.txt

commit 5f9c8b7e5eeb4eadd07ef9adae9adaf9b0c85dc4
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Wed Oct 14 20:46:55 2015 -0400

    converted to txt file, rtf was being weird
```

```
commit 23d0ffc8bae17563caedde534f5000a6aa677049
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Wed Oct 14 20:41:21 2015 -0400

    added ReferenceManual folder

commit e8080df7a4f04e2f1c815d793e6af917697fce7f
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Wed Oct 14 20:38:26 2015 -0400

    added MeetingNotes.txt

commit 8437e3700a5150c23251534466f2fcd9a17fccd1
Author: Rashedul Haydar <rh2712@columbia.edu>
Date:   Wed Oct 14 20:35:46 2015 -0400

    added MeetingNotes.rtf

commit 2b813a4768920b48f359dcfed3783c8f9cfd0729
Author: Tim Waterman <watermantium@gmail.com>
Date:   Mon Oct 12 00:25:18 2015 -0400

    Started matrix mult. Pthread structure is done. Need to
   finish later

commit d536c21195dfd58aec354876d08fa33c469f8b88
Author: Rashedul Haydar <rhaydar8@gmail.com>
Date:   Sun Oct 11 21:24:26 2015 -0400

    Updated First step section

commit d9e2dd310598b609ae5b671838e7201575e431c3
Author: ms4985 <ms4985@columbia.edu>
Date:   Thu Oct 8 18:31:02 2015 -0400

    Create openMP_example_link

commit cd0e4e90156605fd5823f613e808c8ab21425fa3
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Tue Oct 6 14:52:46 2015 -0400
```

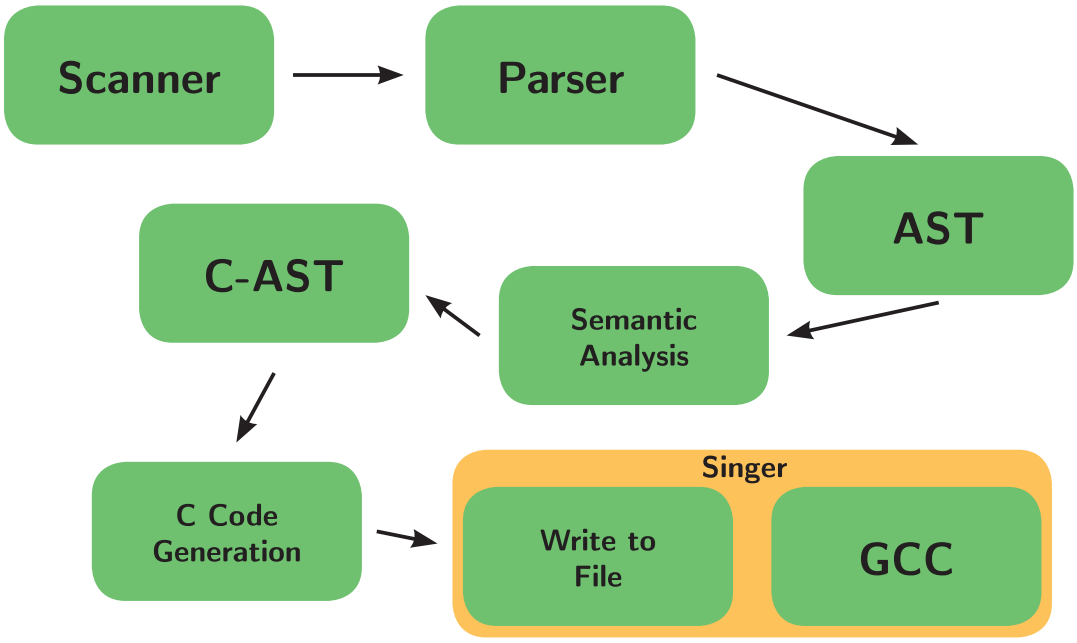
```
    Create meetingNotes.md

commit 28c98405da3a936dc96c30cb28ae1695a7ce265a
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Wed Sep 30 15:49:06 2015 -0400

    remade pdf

commit 096049cfc34a57a32fb1e15c5dda0dbe1050c23f
Author: Rashedul Haydar <rhaydar8@gmail.com>
Date:   Wed Sep 30 15:44:49 2015 -0400

    Corrected typos

commit 2a108e23e604dc41ea2503829faf1893e680958b
Author: Rashedul Haydar <rhaydar8@gmail.com>
Date:   Wed Sep 30 15:43:38 2015 -0400

    Corrected typo and added roles

commit 3af13488ec582d1d724f881e3fa2880850f7e505
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Mon Sep 28 08:41:56 2015 -0400

    updated pdf

commit 5cd818cf0dc19a72e27a3313ce775f56968290f2
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Sep 27 23:24:55 2015 -0400

    Ex3 edited

commit bb8f808bd5b0f476e1367b25cac299a8ec765f85
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Sep 27 17:13:15 2015 -0400

    fixed examples, need example program

commit fd87df4bae7efa64a7559c26b64c4eeae810e2db
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Sep 27 12:39:23 2015 -0400
```

```
    minor changes

commit b2b4242a0aca8edb1b53aa81270aa0595c3cc385
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Sep 27 12:13:58 2015 -0400

    added para RE async keywork

commit 92a02644f4372f4e83d60e3b5814b2069a9ff94d
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Sep 27 12:01:27 2015 -0400

    added code example

commit ae8de78c1fd46f0e97b330a8d2b0ffbfa94732eb
Merge: 1dc3887 10c6456
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Sep 27 11:52:01 2015 -0400

    Merge branch 'master' of https://github.com/danhcole/4115
   _lang

commit 1dc3887de4b592953bd64898d30061ae630d86f9
Author: Tim Waterman <watermantium@gmail.com>
Date:   Sun Sep 27 11:51:49 2015 -0400

    Code snippet added

commit 10c64566f531bf806d9f08c51054fb95d5737a61
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sun Sep 27 11:51:47 2015 -0400

    small changes to .tex

commit f9c5317d8450c748a4cf81ca45cfd28f1a3f1d28
Author: Rashedul Haydar <rhaydar8@gmail.com>
Date:   Sat Sep 26 21:52:32 2015 -0400

    Fixed typos
```

```
commit 6da5e19dfaab76bbd201fb71df4558cf991fa18b
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Sat Sep 26 21:42:24 2015 -0400

    added rough draft of the proposal language

commit e13d614a71d90849f35c1ba97f016a8781adc514
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Wed Sep 16 14:41:38 2015 -0400

    Markdown is hard

commit 813e51459a7a552d72a5e584887e1b93f9199eb6
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Sep 15 12:58:54 2015 -0400

    1st meeting time

commit 30e38166da5a57f2c946d66312df0bb68760033c
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Sep 15 11:52:48 2015 -0400

    updated w/ links to last semesters projects

commit 9ea75aaf3e11cdbeaf823e997ade807c76afe94c
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Sep 15 09:49:18 2015 -0400

    minor update to readme

commit 62b7ea6a2e751c08874c7212167b3a6287b173c5
Author: Daniel Cole <takeitfromthedan@gmail.com>
Date:   Tue Sep 15 09:46:55 2015 -0400

    added link to class page

commit e63c1d0b41a77938f668162dc8ba785e47458a62
Author: Daniel Cole <dhc2131@columbia.edu>
Date:   Tue Sep 15 09:25:17 2015 -0400

    Initial commit
```

Architectural Design

Block Diagram



Interface Description

Scanner

`stch_scanner.mll` — The scanner is written in OCamlLex. It takes input from the source file, and tokenizes it into keywords, identifiers, and literals. It scans over and removes both single line and multi-line comments, as well as all whitespace not in string literals. Any token that is not a keyword, or does not meet the criteria for an identifier or literal will throw a scanner error.

Parser/AST

`stch_parser.mly`

`stch_ast.ml` — The parser is written in OCamlYacc. It takes the tokens from the scanner, and using the grammar defined in the parser and the datatypes defined in the AST, generates an abstract syntax tree. The rules in the parser insure that code that passes this step is syntactically correct, although not necessarily semantically correct. Any error at this stage will throw a parser error. The AST file also contains pretty printing functions for all datatypes defined therein.

Semantic Checking/CAST

`stch_ast.mll`

`stch_semantic.ml`

`stch.cast.mli` — Stitch first runs the AST through the semantic analyzer. This pass insures that the code is semantically valid. The output of the semantic analyzer is another AST, a C language AST. The major difference here is that the CAST carries with it a Stitch Environment, which has, most importantly, a list of declared functions, and a symbol table, which contains all declared identifiers and their types. Stitch's symbol table also contains information on the expression that the identifier references, which is used in the C code generation to build pthread related code.

Code Generation

`c_generator.ml`

The CAST, which has already been semantically analyzed is now pretty printed. The bulk of the work to add multithreading is done here. The C generator performs multiple passes on the CAST. First any non-`main` functions are printed. Then any `stitch` loops are analyzed, and their statements are turned into a function. Finally, `main` is printed. This insures proper scoping of functions, and that all functions declared before main can be called in any Stitch block. To convert `stitch` loops to multithreaded `pthread` code, the C generator first takes the body of the loop and transforms it into a function. The generator also builds a custom `struct` for each `stitch` loop, that contains all in-scope, non-local variables that the loop will need access too. It then generates the `pthread` specific code, as well as a `for` loop that creates and runs the threads. The function containing the code body, as well as the structure containing all needed variables is passed into the `pthread`.

Test Plan

GCD

```
int gcd(int a, int b) {
  while (a != b) {
    if (a > b) {
      a = a - b;
    }
    else {
      b = b - a;
    }
  }
  return a;
}

int main() {
  int x = 1;
  int y = 10;

  int z = gcd(x,y);

  print(z);

  return 0;
}
```

Listing 1: Stitch

GCD

```c
#include "stch_headers.h"


int gcd(int b, int a)
{
while ((a != b)) {
if ((a > b))
{
a = (a - b);
}
else
{
b = (b - a);
}
}
return a;
}




int main()
{
int x = 1;
int y = 10;
int z = gcd(x, y);
printf("%d\n", z);
return 0;
}
```

Listing 2: C

Stitch loop Matrix Multiplication

```
int main() {

  int i = 0;
  int test = 6;

  int a[6][6];
  int k = 0;
  int j = 0;

  for(k = 0; k < 6; k = k + 1) {
    for(j = 0; j < 6; j = j + 1) {
      a[k][j] = 0;
    }
  }

  stitch i from 0 to 6 by 1: {

    int j;
    for(j = 0; j < 6; j = j + 1) {
      a[i][j] = a[i][j] + 10;
    }
  }

    for(j = 0; j < 6; j = j + 1) {
      for(k = 0; k < 6; k = k + 1) {
        print(a[j][k]);
      }
    }

  return 0;

}
```

Listing 3: Stitch

Stitch loop Matrix Multiplication

```
1 #include "stch_headers.h"
2
3
4
5 struct stch_rangeInfo_0 {
6 int begin;
7 int end;
8 int stepSize;
9 int k;
10 int (* a)[6];
11 int test;
12
13
14 };
15
16 void *_0 (void *vars) {
17  int i = 0;
18  for(i = ((struct stch_rangeInfo_0 *)vars)->begin; i < ((struct
      stch_rangeInfo_0 *)vars)->end; i++) {
19 {
20 int j;
21 for (j = 0 ; j < 6 ; j = j + 1) {
22 ((struct stch_rangeInfo_0 *)vars)->a[i][j] = ((struct stch_rangeInfo_0 *)vars
      )->a[i][j] + 10;
23 }
24 }
25
26 }
27 return (void*)0;
28 }
29
30 int main()
31 {
32 int i = 0;
33 int test = 6;
34 int a[6][6];
35 int k = 0;
36 int j = 0;
37 for (k = 0 ; (k < 6) ; k = (k + 1)) {
38 for (j = 0 ; (j < 6) ; j = (j + 1)) {
39 a[k][j] = 0;
40 }
41 }
42
43 pthread_t *threadpool_0 = malloc(NUMTHREADS * sizeof(pthread_t));
44 struct stch_rangeInfo_0 *info_0 = malloc(sizeof(struct stch_rangeInfo_0) *
```

```
    NUMTHREADS);
int thread_0 = 0;
for(i = 0;i < 6;i = i+6/NUMTHREADS) {
info_0[thread_0].begin = i;
info_0[thread_0].k = k;
info_0[thread_0].a = a;
info_0[thread_0].test = test;

if((i + 2*(6/NUMTHREADS)) > 6) {
info_0[thread_0].end = 6;
i = 6;
}
else {
info_0[thread_0].end = i + 6/NUMTHREADS;
}
int e = pthread_create(&threadpool_0[thread_0], NULL, _0, &info_0[thread_0]);
if (e != 0) {
perror("Cannot create thread!");
free(threadpool_0); //error, free the threadpool
exit(1);
}
thread_0++;
}

//loop and wait for all the threads to finish
for(i = 0; i < NUMTHREADS; i++) {
pthread_join(threadpool_0[i], NULL);
}
//now we loop and resolve any accumulators
for(i = 0; i < NUMTHREADS; i++) {

}

for (j = 0 ; (j < 6) ; j = (j + 1)) {
for (k = 0 ; (k < 6) ; k = (k + 1)) {
printf("%d\n", a[j][k]);
}
}
return 0;
}
```

Listing 4: C

Test Suite Log

Full test code supplied in appendix

```
******************
* Positive Tests *
******************
Starting Test ./_tests/accum1.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!


Starting Test ./_tests/arith1.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!


Starting Test ./_tests/arith2.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!


Starting Test ./_tests/array1.stch
==========
COMPILE SUCCESSFUL!
```

```
DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/arrayassign.stch
=======================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/break1.stch
======================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/collatz.stch
======================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/collatz2.stch
======================
COMPILE SUCCESSFUL!
```

```
DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/comment1.stch
====================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/comment3.stch
====================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/escape.stch
====================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/exit1.stch
====================
COMPILE SUCCESSFUL!

DIFFing Output
```

```
==========
TEST SUCCESSFUL!


Starting Test ./_tests/file1.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!


Starting Test ./_tests/file2.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!


Starting Test ./_tests/for1.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!


Starting Test ./_tests/func1.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
```

```
TEST SUCCESSFUL!


Starting Test ./_tests/func2.stch
=================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/func4.stch
=================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/func5.stch
=================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/gcd.stch
=================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!
```

```
Starting Test ./_tests/hello1.stch
==================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/hello2.stch
==================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/if1.stch
==================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/if2.stch
==================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!
```

```
Starting Test ./_tests/if3.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!


Starting Test ./_tests/main.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!


Starting Test ./_tests/matmult.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!


Starting Test ./_tests/matrix1.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!
```

```
Starting Test ./_tests/matrixinit.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!


Starting Test ./_tests/matrixstitch.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!


Starting Test ./_tests/negate.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!


Starting Test ./_tests/ops1.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!
```

```
Starting Test ./_tests/ops2.stch
====================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/sem2.stch
====================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/stitch1.stch
====================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/stitch2.stch
====================
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!


Starting Test ./_tests/stitch3.stch
```

```
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!


Starting Test ./_tests/stitch4.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!


Starting Test ./_tests/stitch5.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!


Starting Test ./_tests/stitch6.stch
==========
COMPILE SUCCESSFUL!

DIFFing Output
==========
TEST SUCCESSFUL!


Starting Test ./_tests/stitch7.stch
==========
```

```
COMPILE SUCCESSFUL!

DIFFing Output
==============
TEST SUCCESSFUL!



******************
* Negative Tests *
******************
Starting Negative Test ./_ntests/arith3.stch
============================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/array2.stch
============================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/array3.stch
============================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/array4.stch
============================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/arrayinit1.stch
============================
TEST SUCCESSFUL!
```

```
Starting Negative Test ./_ntests/arrayinit2.stch
================================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/char1.stch
================================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/comment2.stch
================================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/comment4.stch
================================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/error.stch
================================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/exit2.stch
================================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/file1.stch
================================
TEST SUCCESSFUL!
```

```
Starting Negative Test ./_ntests/float1.stch
==============================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/func1.stch
==============================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/func2.stch
==============================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/globalvar1.stch
==============================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/if1.stch
==============================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/if2.stch
==============================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/matrixinit.stch
==============================
TEST SUCCESSFUL!
```

```
Starting Negative Test ./_ntests/matrixinit2.stch
==========
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/negate2.stch
==========
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/negate3.stch
==========
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/print.stch
==========
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/sem1.stch
==========
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/sem3.stch
==========
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/stitch1.stch
==========
TEST SUCCESSFUL!
```

```
Starting Negative Test ./_ntests/stitch4.stch
==============================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/unfunc.stch
==============================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/vardecl1.stch
==============================
TEST SUCCESSFUL!


Starting Negative Test ./_ntests/void1.stch
==============================
TEST SUCCESSFUL!


Passed 71 / 71 tests
```

Test Description

We started with a basic hello world program and then started adding cases such as basic arithmetic, comparison and logical operations, conditionals, comments, functions etc. Once we were able to get the basics passing the test suite, we moved on to semantic checking, arrays (1D and 2D), file I/O, and stitch loops. We added negative tests for certain features for a more comprehensive testing suite.

Positive Tests

- Basic arithmetic
- Comparison ops
- Logical ops, negate
- Conditional statements
  - Nested variable declarations
  - Nested conditional statements
- Comments- single and multiple lines
- Functions- single, multiple, gcd
- Break, exit
- Type checking
- 1D arrays, initializing and assigning
- 2D arrays (matrices), initializing and assigning
- File I/O
- Stitch Loops
- Matrix multiplication
- Escaped characters
- Accumulators

Negative Tests

- Arithmetic with mismatched types
- Type checking with void
- Negate with floats, chars
- Global variables
- Comments
- Functions
  - Initializing variables as arguments
  - No return type
  - Declaring functions inside functions
  - Calling undeclared functions
- Arrays
  - Initializing, accessing, size parameter as expression
- Matrices
  - Initializing, accessing
- Print/error with wrong type
- File I/O
- Invalid conditionals
- Stitch Loops
  - Undeclared iterator variable
  - No curly braces around statement block

Test Automation

The test suite stch_testSuite.sh first makes the compiler, then iterates through each test in the positive tests folder, calling the tool chain ?Singer?. ?Singer? runs the compiler on the test program, generating the c code, and compiling the c program with the appropriate runtime headers and c libraries. The test suite then checks if the file compiled, and prints the appropriate response to the screen and the log. It compares the difference between the output generated by the executable and the expected output, and prints the result to the screen and to the log. The negative tests are iterated through in a similar way, however the test only passes if the compilation fails. Finally, the test suite cleans up all the target programs, generated output, and executables.

Tests were added by all members of our team as they were needed. The test script is located in the appendix.

Lessons Learned

Rashedul Haydar

For a semester long project it's very important to try to get at least parts of the project done each week. Thankfully, we planned enough to have progress each week on the project. Having the weekly meetings with our advisor really pushed us to get something done every week. Even with the incremental progress, the last two weeks of the project was still crazy.

Megan Skrypek

I learned how important planning and communication is in tackling a large project like this. Having weekly meetings to work on components of the project as well as discuss future plans really helped us manage our time. Working on key components as a team really helped every member understanding the overall flow of the project, instead of only handling individual components. Some advice for future teams would be to start as early as possible, if you get stuck in the beginning it is very difficult to finish on time since each piece of the compiler builds on one another.

Daniel Cole

First and foremost, this was an amazing learning opportunity. Beyond learning the PLT related topics of complier design, I learned group management, how to work on large scale, long term projects, how to integrate code written by multiple people, and how to partition tasks. One of the big challenges was the long scale of the project. Keeping everything moving, and everyone motivated when the deadline was far away was not always easy. The periodic milestones, as well as the weekly checkins help immensely.

Because so much of our language focused on the C code generation, when we got to that point, (from the end of November on), it became much harder to split up the work, as most work at this point was dependent on previous work, and had to be completed in sequence. For instance, matrices needed arrays done first, and the Stitch loop bodies needed the Stitch functions done first. Along with other classwork, this was the biggest issue we had with balancing the workload.

I'd also like to confirm pretty much everything you said at the beginning of the semester, especially regarding OCaml. It wasn't until around the time I finished up the semantic analyzer that I fully appreciated why you had us work in this language. While it won't be my first choice for most projects going forwards, for this type of thing, it is 100% the best language I can imagine.

Tim Waterman

I learned how to set realistic goals and keep going on a project that's an entire semester long. I learned the importance of consistency in regards to weekly meetings and milestones. And I learned how to think in OCaml. This last one is a bit worrying actually, since I can?t seem to turn it off. My advice for future teams is to start early and stay consistent. Keep the progression small but continuous and you?ll have a much better time.

Code

stch_scanner.mll

```
(*
Stitch Scanner
December 2015
Authors: Dan Cole, Rashedul Haydar & Megan Skrypek

*)

{ open Stch_parser }

rule token = parse
  [' ' '\t' '\r' '\n'] { token lexbuf }
  | "//"             { sline_comment lexbuf }
  | "/*"             { block_comment lexbuf }
  | ';'              { SEMI }
  | ':'              { COLON }
  | '''              { SQUOTE }
  | '"'              { DQUOTE }
  | '('              { LPAREN }
  | ')'              { RPAREN }
  | '['              { LSQUARE }
  | ']'              { RSQUARE }
  | '{'              { LBRACE }
  | '}'              { RBRACE }
  | ','              { COMMA }
  | '*'              { TIMES }
  | '/'              { DIVIDE }
  | '+'              { ADD }
  | '-'              { SUBTRACT }
  | '%'              { MOD }
  | '='              { ASSIGN }
  | "=="             { EQUAL }
  | '!'              { NEGATE }
  | "!="             { NE }
  | "&&"             { AND }
  | "||"             { OR }
  | '>'              { GT }
  | ">="             { GE }
  | '<'              { LT }
  | "<="             { LE }
  | "if"             { IF }
  | "else"           { ELSE }
  | "while"          { WHILE }
  | "for"            { FOR }
  | "stitch"         { STITCH }
  | "from"           { FROM }
  | "to"             { TO }
  | "by"             { BY }
  | "break"          { BREAK }
  | "return"         { RETURN }
  | "void"           { TVOID }
  | "int"            { TINT }
  | "float"          { TFLOAT }
  | "char"           { TCHAR }
  | "int_ap"         { TINTAP }
```

```
55  | "int_am"          { TINTAM }
56  | "float_ap"        { TFLOATAP }
57  | "float_am"        { TFLOATAM }
58  | "FILE"          { TFILE }
59  | ['-' '+']?['0' - '9']+ as i_litr          { INT(int_of_string i_litr) }
60  | ['-' '+']?['0'-'9']?'.'['0'-'9']* as f_litr     { FLOAT(float_of_string f_litr)
      }
61  | '''([^''']  as ch_litr)'''            { CHAR(ch_litr)}
62  | '''(['\\']['\\' ''' '"' 't' 'n'] as esc_ch)'''    { ESCAPE(esc_ch)}
63  | '"'([^'"']* as st_litr)'"'            { STRING(st_litr)}
64  | ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']* as litr { ID(litr) }
65  | eof { EOF }
66  | _ as char { raise (Failure("illegal character " ^ Char.escaped char))}
67
68  and sline_comment = parse
69    '\n'          { token lexbuf }
70    | _           { sline_comment lexbuf }
71
72  and block_comment = parse
73    "*/"          { token lexbuf }
74    | _           { block_comment lexbuf }
```

stch_parser.mly

```
1
2 %{ open Stch_ast %}
3
4 %token SEMI SQUOTE DQUOTE COLON LPAREN RPAREN LSQUARE RSQUARE LBRACE RBRACE
5 %token COMMA TIMES DIVIDE ADD SUBTRACT MOD
6 %token ASSIGN EQUAL NEGATE NE
7 %token AND OR
8 %token GT GE LT LE
9 %token FROM TO BY
10 %token IF ELSE WHILE FOR STITCH BREAK RETURN TVOID TINT TFLOAT TCHAR TINTAP TINTAM
    TFLOATAP TFLOATAM TFILE
11 %token VOID
12 %token <int> INT
13 %token <char> CHAR
14 %token <string> ESCAPE
15 %token <float> FLOAT
16 %token <string> STRING
17 %token <string> ID
18 %token EOF
19
20 %nonassoc NOELSE
21 %nonassoc ELSE
22 %right ASSIGN
23 %left OR
24 %left AND
25 %left EQUAL NE
26 %left LT GT LE GE
27 %left ADD SUBTRACT
28 %left TIMES DIVIDE MOD
29 %right NEGATE
30
31 %start program
32 %type <Stch_ast.program> program
33
34 %%
35
36 program:
37   /*decls EOF {$1}*/     { [], [] }
38   | program stmt    SEMI { ($2 :: fst $1), snd $1 }
39   | program fdecl      { fst $1, ($2 :: snd $1) }
40
41 fdecl:
42   type_name ID LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE
43   { { fdecl_type = $1;
44     fdecl_name = $2;
45     fdecl_formals = $4;
46     body = List.rev $7; } }
47
48 type_name:
49   TINT       { Tint }
50 | TFLOAT     { Tfloat }
51 | TCHAR      { Tchar }
52 | TVOID      { Tvoid }
53 | TINTAP     { Tintap }
```

```
54 | TINTAM    { Tintam }
55 | TFLOATAP    { Tfloatap }
56 | TFLOATAM    { Tfloatam }
57 | TFILE     { Tfile }
58
59 formals_opt:
60   /* nothing */   { [] }
61 | formal_list   { List.rev $1 }
62
63 formal_list:
64   vdecl           { [$1] }
65 | formal_list COMMA vdecl { $3 :: $1 }
66
67 vdecl:
68   type_name ID
69   {{
70     vdecl_type  = $1;
71     vdecl_name  = $2;
72   }}
73
74 arraydecl:
75   type_name ID LSQUARE expr_opt RSQUARE
76   {{
77     arraydecl_type = $1;
78     arraydecl_name = $2;
79     arraydecl_size = $4;
80   }}
81
82 matrixdecl: /* two dimensional array implementation */
83   type_name ID LSQUARE expr_opt RSQUARE LSQUARE expr_opt RSQUARE
84   {{
85     matrixdecl_type = $1;
86     matrixdecl_name = $2;
87     matrixdecl_rows = $4;
88     matrixdecl_cols = $7;
89   }}
90
91
92 stmt_list:
93 /*nothing*/     { [] }
94 | stmt_list stmt  { $2 :: $1 }
95
96 stmt:
97   expr SEMI                    { Expr($1) }
98 | vdecl SEMI                   { Vdecl($1) }
99 /* One dimensional array stuff */
100 | arraydecl SEMI                 { ArrayDecl($1) }
101 | arraydecl ASSIGN LBRACE actuals_opt RBRACE SEMI
102   { ArrayInit($1, $4) }
103 /* Two dimensional array statements */
104 | matrixdecl SEMI               { MatrixDecl($1) }
105 | matrixdecl ASSIGN LBRACE matrix_rev_list RBRACE SEMI
106   { MatrixInit($1, $4) }
107 | RETURN expr_opt SEMI              { Return($2) }
108 | LBRACE stmt_list RBRACE             { Block(List.rev $2) }
109 | IF LPAREN expr RPAREN stmt %prec NOELSE   { If($3, $5, Block([])) }
110 | IF LPAREN expr RPAREN stmt ELSE stmt      { If($3, $5, $7) }
111 | FOR LPAREN expr_opt SEMI expr_opt SEMI expr_opt RPAREN stmt
```

```
112   { For($3,$5,$7,$9) }
113 | WHILE LPAREN expr RPAREN stmt          { While($3, $5) }
114 | STITCH expr FROM expr TO expr BY expr COLON stmt
115   { Stitch($2,$4,$6,$8,$10) }
116 | vdecl ASSIGN expr SEMI              { Assign($1, $3) }
117 | BREAK SEMI                    { Break }
118
119 expr_opt:
120   /*nothing*/ { Noexpr }
121 | expr       { $1 }
122
123 expr:
124 /*Primitives*/
125 | INT   { Int($1) }
126 | FLOAT   { Float($1) }
127 | CHAR    { Char($1) }
128 | ESCAPE  { Escape($1)}
129 | ID    { Id($1) }
130 | STRING  { String($1) }
131 /*Array*/
132 | ID LSQUARE expr RSQUARE ASSIGN expr     { Array_Item_Assign($1, $3, $6) }
133 | ID LSQUARE expr RSQUARE         { Array_Index_Access($1, $3) }
134 /*Matrix */
135 | ID LSQUARE expr RSQUARE LSQUARE expr RSQUARE ASSIGN expr
136   { Matrix_Item_Assign($1, $3, $6, $9) }
137 | ID LSQUARE expr RSQUARE LSQUARE expr RSQUARE
138   { Matrix_Index_Access($1, $3, $6) }
139 /*Arithmetic*/
140 | expr ADD    expr  { Binop($1, Add, $3) }
141 | expr SUBTRACT expr  { Binop($1, Subtract, $3) }
142 | expr TIMES    expr  { Binop($1, Times, $3) }
143 | expr DIVIDE   expr  { Binop($1, Divide, $3) }
144 | expr MOD    expr  { Binop($1, Mod, $3) }
145 /*Comparison*/
146 | expr EQUAL expr   { Binop($1, Equal, $3) }
147 | expr NE    expr   { Binop($1, Ne, $3) }
148 | expr LT    expr   { Binop($1, Lt, $3) }
149 | expr LE    expr   { Binop($1, Le, $3) }
150 | expr GT    expr   { Binop($1, Gt, $3) }
151 | expr GE    expr   { Binop($1, Ge, $3) }
152 /*Logical*/
153 | expr OR expr      { Binop($1, Or, $3) }
154 | expr AND expr     { Binop($1, And, $3) }
155 /*Unary*/
156 | NEGATE expr     { Negate($2)}
157 /*Miscellanenous*/
158 | ID LPAREN actuals_opt RPAREN { Call($1, $3) }
159 | LPAREN expr RPAREN  { $2 }
160 | ID ASSIGN expr    { Assign2($1, $3) }
161
162 /*List items for matrix initialization*/
163 matrix_rev_list:
164   matrix_list   { List.rev $1 }
165
166 matrix_list:
167   LBRACE actuals_opt RBRACE   { [$2] }
168 | matrix_list COMMA LBRACE actuals_opt RBRACE { $4::$1 }
169
```

```
170 actuals_opt:
171   /*nothing*/ { [] }
172 | actuals_list  { List.rev $1 }
173
174 actuals_list:
175   expr            { [$1] }
176 | actuals_list COMMA expr {$3 :: $1 }
```

stch_ast.ml

```
1 (*
2 Stitch AST
3 December 2015
4 Authors: Dan Cole, Rashedul Haydar, Tim Waterman, & Megan Skrypek
5
6 Our Stitch Abstract Syntax Tree
7 *)
8
9 type op = Add | Subtract | Times | Divide | Mod | Equal | Ne | Lt | Le | Gt | Ge
10          | Or | And
11 type dataType = Tint | Tfloat | Tchar | Tvoid | Tstring | Tintap | Tintam | Tfloatap
       | Tfloatam | Tfile
12
13 type vdecl = {
14   vdecl_type      : dataType;
15   vdecl_name      : string;
16 }
17
18 (* Expressions *)
19 type expr =
20     Int of int
21   | Float of float
22   | Char of char
23   | Escape of string
24   | Id of string
25   | String of string
26   | Binop of expr * op * expr
27   | Negate of expr
28   | Call of string * expr list
29   | Assign2 of string * expr
30   | Array_Item_Assign of string * expr * expr
31   | Array_Index_Access of string * expr
32   | Matrix_Item_Assign of string * expr * expr * expr
33   | Matrix_Index_Access of string * expr * expr
34   | Access of string * string
35   | Noexpr
36
37 (* Array Declarations *)
38 type arraydecl = {
39     arraydecl_type : dataType;
40     arraydecl_name : string;
41     arraydecl_size : expr;
42
43   }
44
45 (* Matrix Declarations *)
46   type matrixdecl = {
47     matrixdecl_type : dataType;
48     matrixdecl_name : string;
49     matrixdecl_rows : expr;
50     matrixdecl_cols : expr;
51
52   }
53
```

```
54 (* Statements *)
55 type stmt =
56     Block of stmt list
57   | Vdecl of vdecl
58   | Expr of expr
59   | Return of expr
60   | If of expr * stmt * stmt
61   | For of expr * expr * expr * stmt
62   | While of expr * stmt
63   | Stitch of expr * expr * expr * expr * stmt
64   | Assign of vdecl * expr
65   | ArrayDecl of arraydecl
66   | ArrayInit of arraydecl * expr list
67   | MatrixDecl of matrixdecl
68   | MatrixInit of matrixdecl * expr list list
69   | Break
70
71 type fdecl = {
72     fdecl_type : dataType;
73     fdecl_name : string;
74     fdecl_formals : vdecl list;
75     body : stmt list;
76   }
77
78 type program = stmt list * fdecl list
79
80 (* Pretty printer, used for testing, and in parts of the c_generator *)
81 let string_of_dataType = function
82   Tint -> "int"
83   | Tfloat -> "float"
84   | Tchar -> "char"
85   | Tvoid -> "void"
86   | Tstring -> "char *"
87   | Tintap -> "int"
88   | Tintam -> "int"
89   | Tfloatap -> "float"
90   | Tfloatam -> "float"
91   | Tfile -> "FILE *"
92
93 let rec string_of_expr = function
94     Int(l) -> string_of_int l
95   | Float(l) -> string_of_float l
96   | Char(l) ->  "\'" ^ String.make 1 l ^ "\'"
97   | Escape(l) -> "\'" ^ l ^ "\'"
98   | Id(s) -> s
99   | String(s) -> "\"" ^ s ^ "\""
100  | Binop(e1, o, e2) ->
101      string_of_expr e1 ^ " " ^
102      (match o with
103   Add -> "+" | Subtract -> "-" | Times -> "*" | Divide -> "/"
104      | Equal -> "==" | Ne -> "!="
105      | Lt -> "<" | Le -> "<=" | Gt -> ">" | Ge -> ">="
106      | Or -> "||" | And -> "&&" | Mod -> "%" ) ^ " " ^
107      string_of_expr e2
108  | Negate(e) -> "!" ^ string_of_expr e
109  | Call(f, el) -> (match f with "print" -> "printf" | _ -> f) ^ "(" ^ String.concat
       ", " (List.map string_of_expr el) ^ ")"
110  | Assign2(i, e) -> i ^ " = " ^ string_of_expr e ^ ";\n"
```

```
111   | Array_Item_Assign(id, ind, e) -> id ^ "[" ^ string_of_expr ind ^"] = " ^
        string_of_expr e
112   | Array_Index_Access(id, index) -> id ^ "[" ^ string_of_expr index ^ "]"
113   | Matrix_Item_Assign(id, row, col, ex) -> id ^ "[" ^ string_of_expr row ^ "][" ^
        string_of_expr col ^ "] = " ^ string_of_expr ex
114   | Matrix_Index_Access(id, row, col) -> id ^ "[" ^ string_of_expr row ^ "][" ^
        string_of_expr col ^ "]"
115   | Access(f, s) -> f ^ "." ^ s
116   | Noexpr -> ""
117
118 let string_of_vdecl vdecl = string_of_dataType vdecl.vdecl_type ^ " " ^ vdecl.
      vdecl_name
119
120 let string_of_arraydecl arraydecl = string_of_dataType arraydecl.arraydecl_type ^ "
      " ^ arraydecl.arraydecl_name ^ "[" ^
121     string_of_expr arraydecl.arraydecl_size ^ "]"
122
123 let string_of_matrixdecl m = string_of_dataType m.matrixdecl_type ^ " " ^ m.
      matrixdecl_name ^ "[" ^
124     string_of_expr m.matrixdecl_rows ^ "][" ^ string_of_expr m.matrixdecl_cols ^ "]"
125
126 let string_of_arraylist el = "{" ^ String.concat ", " (List.map string_of_expr el) ^
      "}"
127
128 let rec string_of_matrixlist (seed: string) el = match el with
129     [] -> seed ^ "}"
130     | head::tail -> string_of_matrixlist (seed ^ string_of_arraylist head ^ ",\n")
          tail
131
132 let rec string_of_stmt = function
133     Block(stmts) ->
134       "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
135   | Expr(expr) -> string_of_expr expr ^ ";\n";
136   | Vdecl(v) -> string_of_dataType v.vdecl_type ^ " " ^ v.vdecl_name ^ ";\n";
137   | Return(expr) -> "return " ^ string_of_expr expr ^ ";\n";
138   | If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s
139   | If(e, s1, s2) ->  "if (" ^ string_of_expr e ^ ")\n" ^
140       string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2
141   | For(e1, e2, e3, s) ->
142       "for (" ^ string_of_expr e1  ^ " ; " ^ string_of_expr e2 ^ " ; " ^
143       string_of_expr e3  ^ ") " ^ string_of_stmt s
144   | While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^ string_of_stmt s
145   | Stitch(e1,e2,e3,e4,s) ->
146       "stitch " ^ string_of_expr e1 ^ " from " ^ string_of_expr e2 ^ " to " ^
147         string_of_expr e3 ^ " by " ^ string_of_expr e4 ^ " : " ^ string_of_stmt s
148   | Assign(v, e) -> string_of_vdecl v ^ " = " ^ string_of_expr e ^ ";\n"
149   | ArrayDecl(a) -> string_of_arraydecl a ^ ";\n"
150   | ArrayInit(arraydecl, el) -> string_of_arraydecl arraydecl ^ " = " ^
        string_of_arraylist el ^ ";\n"
151   | MatrixDecl(m) -> string_of_matrixdecl m ^ ";\n"
152   | MatrixInit(mdecl, li) -> string_of_matrixdecl mdecl ^ " = " ^
        string_of_matrixlist "{" li ^ ";\n"
153   | Break -> "break;"
154
155 let string_of_fdecl fdecl =
156   string_of_dataType fdecl.fdecl_type ^ " " ^ fdecl.fdecl_name ^ "(" ^ String.concat
        ", " (List.map string_of_vdecl fdecl.fdecl_formals) ^ ")\n{\n" ^
157   String.concat "" (List.map string_of_stmt fdecl.body) ^
```

```
158   "}\n"
159
160 let string_of_program (stmts, funcs) =
161   String.concat "" (List.map string_of_stmt stmts) ^ "\n" ^
162   String.concat "\n" (List.map string_of_fdecl funcs)
```

stch_semantic.ml

```
1 (*
2 Semantic Analyzer
3 December 2015
4 Authors: Dan Cole & Tim Waterman
5
6 Takes the AST and runs semantic analysis on it, turning it into
7 a C_AST
8 *)
9
10 open Stch_ast
11 open Stch_cast
12 exception Error of string
13
14 (* Globals for procedurally generating suffix for stitch items *)
15 type stch_name_gen = { mutable name : int }
16 let sn = {name = 0;}
17
18 (* symbol table -> string *)
19 let string_of_symTable (syms: symTable) = let str = "SymTable: \n" ^
20         String.concat "\n" (List.map (fun (typ, name, _) -> "[" ^
21         Stch_ast.string_of_dataType typ ^ " " ^ name ^ "]") syms.vars) ^ "\n"
22   in print_endline str
23
24 (* find a variable (and associated type) in the symbol table *)
25 let rec find_variable (scope: symTable) name =
26   try
27     List.find (fun (_, s, _) -> s = name ) scope.vars
28   with Not_found -> match scope.parent with
29   Some(parent) -> find_variable parent name
30   | _ -> raise (Error("Bad ID " ^ name)) (* in general, any type mismatch raises an
       error *)
31
32 (* check to see if a function has been defined *)
33 let rec find_func (funcs: c_fdecl list) fname =
34   try
35     List.find ( fun fn -> fn.fdecl_name = fname ) funcs
36   with Not_found -> raise (Error ("Function call not recognized: " ^ fname))
37
38 (* type check binary operations *)
39 (* for now, Stitch does not support type coercion, so binops must be int/int or flt/
       flt  *)
40 let check_binop (lhs: dataType) (rhs: dataType) (env: stch_env) : (Stch_ast.dataType
       ) =
41   match (lhs, rhs) with
42     (Tint, Tint)   -> Tint
43   | (Tfloat, Tfloat) -> Tfloat
44   | (_, _) -> raise (Error("Incompatable data types for binop"))
45
46 (* check variable decleration, returns a C_Vdecl *)
47 let check_vdecl (decl: vdecl) (env: stch_env) =
48   let invalid = List.exists (fun (_, s, _) -> s = decl.vdecl_name) env.scope.vars in
49     if invalid then
50       raise (Error("Variable already declared"))
51     else
```

```
52      env.scope.vars <- (decl.vdecl_type, decl.vdecl_name, C_Noexpr)::env.scope.vars
            ;
53      let v = { Stch_cast.vdecl_type = decl.vdecl_type;
54            Stch_cast.vdecl_name = decl.vdecl_name } in
55        C_Vdecl(v)
56
57 (* same as check_vdecl, except that it returns a triple of vdecl, datatype, name *)
58 let check_vdecl_t (decl: vdecl) (env: stch_env) =
59   let invalid = List.exists (fun (_, s, _) -> s = decl.vdecl_name) env.scope.vars in
60     if invalid then
61       raise (Error("Variable already declared"))
62     else
63       env.scope.vars <- (decl.vdecl_type, decl.vdecl_name, C_Noexpr)::env.scope.vars
            ;
64       let v = { Stch_cast.vdecl_type = decl.vdecl_type;
65             Stch_cast.vdecl_name = decl.vdecl_name } in
66         v, v.vdecl_type, v.vdecl_name
67
68 (* type check an expression and put into c_ast *)
69 let rec check_expr (e: expr) (env: stch_env) : (Stch_cast.c_expr * Stch_ast.dataType
    ) =
70   match e with
71   (* primitives get a free pass *)
72     Int(l)  -> C_Int(l), Tint
73   | Float(l)  -> C_Float(l), Tfloat
74   | Char(l)   -> C_Char(l), Tchar
75   | Escape(l) -> C_Escape(l), Tchar
76   | String(l) -> C_String(l), Tstring
77   (* For ID's, check to see if the variable has been declared, if it has, get the
        name and type *)
78   | Id(l) ->
79     let var = try find_variable env.scope l
80       with Not_found -> raise(Error("Undefined Identifier" ^ l))
81         in
82         let (typ, vname, _) = var in
83         C_Id(vname, typ), typ
84   (* other exprs need to call their respective check functions *)
85   | Binop(lhs, o, rhs) -> binop_ret lhs o rhs env
86   | Negate(l) -> check_negate l env
87   | Call(f, b) -> check_call f b env
88   | Assign2(lhs, rhs) -> check_assign2 lhs rhs env
89   | Array_Index_Access(name, index) -> check_array_index name index env
90   | Array_Item_Assign(name, index, ex) -> check_array_item_assign name index ex env
91   | Matrix_Index_Access(name, row, col) -> check_matrix_index name row col env
92   | Matrix_Item_Assign(name, row, col, ex) -> check_matrix_item_assign name row col
        ex env
93   | Noexpr -> C_Noexpr, Tvoid
94   | _ -> C_Noexpr, Tvoid  (* Can remove when everything else is added *)
95
96   (* check negation.  As of now, only ints and floats can be negated *)
97   and check_negate (e: expr) (env: stch_env) =
98     let exp = check_expr e env in
99     match snd exp with
100      Tint -> C_Negate((fst exp)), Tint
101    | Tfloat -> C_Negate((fst exp)), Tfloat
102    | _ -> raise (Error("Cannot negate type " ^ string_of_dataType (snd exp)))
103
104  (* check the binop return type*)
```

```
105  and binop_ret (lhs: expr) (o: op) (rhs: expr) (env: stch_env) : (Stch_cast.c_expr
         * Stch_ast.dataType) =
106    let (lhs, t1) = check_expr lhs env
107    and (rhs, t2) = check_expr rhs env in
108
109    match o with
110      Add -> C_Binop(lhs, o, rhs), check_binop t1 t2 env
111    | Subtract -> C_Binop(lhs, o, rhs), check_binop t1 t2 env
112    | Times -> C_Binop(lhs, o, rhs), check_binop t1 t2 env
113    | Divide -> C_Binop(lhs, o, rhs), check_binop t1 t2 env
114    | Mod -> C_Binop(lhs, o, rhs), check_binop t1 t2 env
115    | Equal -> C_Binop(lhs, o, rhs), check_binop t1 t2 env
116    | Ne -> C_Binop(lhs, o, rhs), check_binop t1 t2 env
117    | Lt -> C_Binop(lhs, o, rhs), check_binop t1 t2 env
118    | Le -> C_Binop(lhs, o, rhs), check_binop t1 t2 env
119    | Gt -> C_Binop(lhs, o, rhs), check_binop t1 t2 env
120    | Ge -> C_Binop(lhs, o, rhs), check_binop t1 t2 env
121    | Or -> C_Binop(lhs, o, rhs), check_binop t1 t2 env
122    | And -> C_Binop(lhs, o, rhs), check_binop t1 t2 env
123
124  (* check assign2 (i.e. expr assign) *)
125  and check_assign2 (lhs: string) (rhs: expr) (env: stch_env) : (Stch_cast.c_expr *
         Stch_ast.dataType) =
126    let (t1, _, _) = find_variable env.scope lhs
127    and (rhs, t2) = check_expr rhs env in
128    if t1 = t2 || (t1 = Tintap && t2 = Tint) then
129      C_Assign2(lhs, rhs), t2
130    else if t1 = Tint && t2 = Tchar then
131      C_Assign2(lhs, rhs), t1
132    else
133      raise (Error("Type mismatch on variable assignment " ^ lhs ^
134        "\nExpected: " ^ string_of_dataType t1 ^ " Got: " ^ string_of_dataType t2))
135
136  (* Checking array access by index. Index should be an int, we just need to make
         sure that the
137    array exists. We could also conceviably rewrite this later to do bounds checking
         *)
138  and check_array_index (n: string) (index: expr) (env: stch_env) =
139    let var = find_variable env.scope n in
140    let (typ, vname, _) = var in
141    let (e, t) = check_expr index env in match t with
142      Tint -> C_Array_Index(vname, e, typ), typ
143      | _ -> raise(Error("Cannot index into an array with type " ^
144          string_of_dataType t))
145
146  (*  Checking matrix access by indices. They should both be ints, row and col.
147    Also we need to check that the variable exists first
148     *)
149  and check_matrix_index (n: string) (row: expr) (col: expr) (env: stch_env) =
150    let var = find_variable env.scope n in
151    let (typ, vname, _) = var in
152    let (erow, trow) = check_expr row env in
153    let (ecol, tcol) = check_expr col env in match (trow, tcol) with
154      (Tint, Tint) -> C_Matrix_Index(vname, erow, ecol, typ), typ
155      | _ -> raise(Error("Cannot index into an array with types " ^ string_of_expr
          row ^ ", " ^
156            string_of_expr col))
```

```
157
158   (* Checking the array assignment to a specific index. Will validate the lhs as a
          valid access, and
159     then will make sure the rhs has the proper type for assignment
160   *)
161   and check_array_item_assign (name: string) (index: expr) (rhs: expr) (env:
          stch_env) =
162     let var = find_variable env.scope name in
163     let (typ, vname, _) = var in
164     let (e, t) = check_expr index env in
165       if t <> Tint then
166         raise(Error("Cannot index into an array with type " ^ string_of_dataType t))
167       else
168         let (erhs, trhs) = check_expr rhs env in
169         (* Hacky for now, allowing anything to store into a int or char array *)
170           if trhs <> typ && (typ <> Tint && typ <> Tchar )then
171             raise(Error("Type mismatch on array item assignment"))
172           else
173             C_Array_Item_Assign(vname, e, erhs), typ
174
175   and check_matrix_item_assign (name: string) (row: expr) (col: expr) (rhs: expr) (
          env: stch_env) =
176     let var = find_variable env.scope name in
177     let (vtyp, vname, _) = var in
178     let (erow, trow) = check_expr row env in
179     let (ecol, tcol) = check_expr col env in
180       if trow <> Tint || tcol <> Tint then
181         raise(Error("Cannot index into a matrix with non-int values"))
182       else
183         let (erhs, trhs) = check_expr rhs env in
184           if trhs <> vtyp then
185             raise(Error("Type mismatch on matrix item assignment"))
186           else
187             C_Matrix_Item_Assign(vname, erow, ecol, erhs), vtyp
188
189   (* check function call *)
190   and check_call (f: string) (el: expr list) (env: stch_env) =
191     let l_expr_typ = List.map (fun e -> check_expr e env) el in
192     let func_ret = find_func env.funcs f in
193     let args_l = find_func_sig f l_expr_typ func_ret in
194       C_Call(func_ret.fdecl_name, args_l), func_ret.fdecl_type
195
196   (* function signature verify *)
197   and find_func_sig (f: string) (opts: (c_expr * dataType) list) (func_ret: c_fdecl)
          = match f with
198     (* special handling for built-in functions
199        Not all built-ins need this (eg exit() ) *)
200      "print" -> (let arg = List.hd opts in
201             match (snd arg) with
202               | Tint -> (fst arg)::[]
203               | Tfloat -> (fst arg)::[]
204               | Tchar -> (fst arg)::[]
205               | Tstring -> (fst arg)::[]
206               | Tintap -> (fst arg)::[]
207               | Tintam -> (fst arg)::[]
208               | _ -> raise (Error("Invalid print type: " ^ string_of_dataType (snd
                      arg))))
209     | "error" -> (let arg = List.hd opts in
```

```
210             match (snd arg) with
211               | Tint -> (fst arg)::[]
212               | Tfloat -> (fst arg)::[]
213               | Tchar -> (fst arg)::[]
214               | Tstring -> (fst arg)::[]
215               | _ -> raise (Error("Invalid error type: " ^ string_of_dataType (snd
                      arg))))
216     (* All other functions *)
217     | _ -> try
218         let formals = func_ret.fdecl_formals in
219           let cexpr = List.map2 (fun (opt: c_expr * dataType) (formal: c_vdecl) ->
220             let opt_typ = snd opt in
221             let formal_type = formal.vdecl_type in
222               if opt_typ = formal_type then
223                 fst opt
224               else
225                 C_Noexpr) opts formals in
226             let matched = List.exists (fun e -> e = C_Noexpr) cexpr in
227             if matched then
228               find_func_sig f opts func_ret
229             else
230               cexpr
231       with Invalid_argument(x) ->
232         raise (Error("Wrong number of args in function call " ^ f))
233
234 (* Helper function for array initialization. This function will recursively traverse
       a list of
235   expressions and try to type match them with the type of the array they're being
        added into.
236   This function is called from check_array_init further down in the code
237 *)
238 let rec check_init_vals (name: arraydecl) (el: expr list) (t: dataType) (env:
      stch_env) =
239   match el with
240     | [] -> name
241     | head::tail -> let (ex, typ) = check_expr head env in
242       if typ = t then
243         check_init_vals name tail typ env
244       else
245         raise(Error("Types of array initialization do not match"))
246
247 (* Checking the types for matrix initialization *)
248 let rec check_matrix_rows (name: matrixdecl) (el: expr list) (t: dataType) (env:
      stch_env) =
249   match el with
250     | [] -> name
251     | head::tail -> let (exp, typ) = check_expr head env in
252       if typ = t then begin
253         check_matrix_rows name tail typ env
254       end
255       else
256         raise(Error("Types of matrix init do not match"))
257
258 (* Check that all the matrix rows are the proper length *)
259 let rec check_matrix_vals (name: matrixdecl) (el: expr list list) (ncols: int) (t:
      dataType) (env: stch_env) =
260   match el with
261     | [] -> name
```

```
262     | head::tail ->
263       if ncols <> List.length head then begin
264         raise(Error("Rows are not matching length in matrix decl"))
265       end
266       else
267         let m = check_matrix_rows name head t env in
268         check_matrix_vals m tail ncols t env
269
270 (* Generate the names for the struct and the anonymous pthread functions *)
271 let gen_name (sn : stch_name_gen) =
272   let i = sn.name in
273     sn.name <- i+1; "_" ^ string_of_int i
274
275 let get_id_from_expr (ex: expr) = match ex with
276   Id(l) -> l
277   |_ -> "_null"
278
279 (* typecheck a statement *)
280 let rec check_stmt (s: Stch_ast.stmt) (env: stch_env) = match s with
281   Block(ss) ->
282     let scope' = { parent = Some(env.scope); vars = []; } in
283       let env' = { env with scope = scope' } in
284       let ss = List.map (fun s -> check_stmt s env') ss in
285       scope'.vars <- List.rev scope'.vars;
286       C_Block(scope', ss)
287   | Vdecl(v) -> check_vdecl v env
288   | Expr(e) -> let (e,t) = check_expr e env in C_Expr(t, e)
289   | ArrayDecl(a) -> check_array_decl a env
290   | ArrayInit(a, el) -> check_array_init a el env
291   | MatrixDecl(m) -> check_matrix_decl m env
292   | MatrixInit(mdecl, el) -> check_matrix_init mdecl el env
293   | Return(e) -> check_return e env
294   | If(e, s1, s2) -> check_if e s1 s2 env
295   | For(e1, e2, e3, s) -> check_for e1 e2 e3 s env
296   | While(e, s) -> check_while e s env
297   | Stitch(e1, e2, e3, e4, s) -> check_stitch e1 e2 e3 e4 s env
298   (* stmt assign needs to be fixed *)
299   | Assign(v, e) -> check_assign v e env
300   | Break -> C_Break
301
302   (* check assign (i.e. stmt assign) *)
303   and check_assign (lhs: vdecl) (rhs: expr) (env: stch_env) =
304     let (v, t1, _) = check_vdecl_t lhs env
305       and (rhs, t2) = check_expr rhs env in
306     if t1 = t2 || (t1 = Tintap && t2 = Tint) then
307       C_Assign(v, rhs)
308   else
309     raise (Error("Type mismatch on variable assignment " ^ string_of_vdecl lhs))
310
311   (* typecheck return (not return type, but keyword 'return') *)
312   and check_return (e: expr) (env: stch_env) =
313     if env.in_func then
314     let (e,t) = check_expr e env in
315       if t = env.retType then
316         C_Return(t, e)
317       else
318         raise (Error("Incompatable return type. Expected type " ^
319           string_of_dataType env.retType ^
```

```
320           ", found type " ^
321           string_of_dataType t))
322     else
323       raise (Error("Invalid 'return' call"))
324
325   and check_array_decl (a: arraydecl) (env : stch_env) =
326
327     (* create a variable declaration out of the array declaration so we can check
          for it *)
328     let ve = { Stch_ast.vdecl_type = a.arraydecl_type;
329           Stch_ast.vdecl_name = a.arraydecl_name} in
330
331     (* check to see if the variable is not already declared *)
332     let invalid = List.exists (fun (_, s, _) -> s = ve.vdecl_name) env.scope.vars in
333     if invalid then
334       raise (Error("Variable " ^ ve.vdecl_name ^ " already declared"))
335     else
336     (* if it isn't, put it in the scope, and make a new c_arraydecl
337       after you typematch the size expression *)
338
339     (* If we have an arraydecl, we want the C_EXPR in the symtable to be an index
          operation, so we can
340       get the size information when we are passing the symtable to the code
          generator
341       This is a bit hacky, but it should work for what we need it to
342     *)
343       let (ex, ty) = check_expr a.arraydecl_size env in
344       env.scope.vars <- (ve.vdecl_type, ve.vdecl_name, C_Array_Index(ve.vdecl_name,
          ex, ve.vdecl_type))::env.scope.vars;
345       let (ex, typ) = check_expr a.arraydecl_size env in
346       match typ with
347         Tfloat -> raise (Error("Invalid array size type, expects int"))
348         | Tchar -> raise (Error("Invalid array size type, expects int"))
349         | Tstring -> raise (Error("Invalid array size type, expects int"))
350         | Tvoid -> raise (Error("Invalid array size type, expects int"))
351       (* else it's a void or an int, and it's allowed *)
352         | _ -> let v = { Stch_cast.arraydecl_type = ve.vdecl_type;
353                 Stch_cast.arraydecl_name = ve.vdecl_name;
354                 Stch_cast.arraydecl_size = a.arraydecl_size} in C_ArrayDecl(v)
355
356   (* checking the array initialization. This will be done in 3 steps
357     1. Check to see if the array can be declared as a new variable
358     2. Make sure that all the args in the list are the same type
359     3. Make sure that the type in the list matches the type
360     4. Make sure that the size of the list matches the size of the decl (low
          priority for now)
361   *)
362   and check_array_init (a: arraydecl) (el: expr list) (env: stch_env) =
363     (* first step: check that we have a valid array decl *)
364     let invalid = List.exists (fun (_,s,_) -> s = a.arraydecl_name) env.scope.vars
          in
365       if invalid then
366         raise (Error("Variable " ^ a.arraydecl_name ^ " already declared"))
367       else begin
368         let (ex, ty) = check_expr a.arraydecl_size env in
369         env.scope.vars <- (a.arraydecl_type, a.arraydecl_name,
370           C_Array_Index(a.arraydecl_name, ex, a.arraydecl_type))::env.scope.vars;
371
```

```
372         (* now that we know it's valid, check the types of the list *)
373         let s = a.arraydecl_size in
374         let i = string_of_expr s in
375         let typ = a.arraydecl_type in
376         (* try to match the init size with the list size.
377            Init size must be an int constant, by C rules *)
378         try
379         if int_of_string i = List.length el then
380         let ret = check_init_vals a el typ env in
381           if ret = a then
382             C_ArrayInit({Stch_cast.arraydecl_name = a.arraydecl_name;
383                    Stch_cast.arraydecl_type = a.arraydecl_type;
384                    Stch_cast.arraydecl_size = a.arraydecl_size;}, el)
385           else
386             raise(Error("Error parsing the list of array init args"))
387         else
388           raise(Error("Size mismatch in array initialization"))
389         with
390         | _ -> raise(Error("Cannot initialize array with a variable"))
391     end
392   and check_matrix_init (m: matrixdecl) (el: expr list list) (env: stch_env) =
393     (* First, we need to check that we have a valid declaration by checking for
          vdecl_t *)
394     (* Check the size of the cols and rows, make sure they match the list counts
395       rows = total # of sublists
396       cols = length of the sublists (must be all the same length)
397     *)
398
399     let invalid = List.exists (fun (_,s,_) -> s = m.matrixdecl_name) env.scope.vars
          in
400       if invalid then
401         raise (Error("Variable " ^ m.matrixdecl_name ^ " already declared"))
402       else begin
403         let (exr, ty) = check_expr m.matrixdecl_rows env in
404         let (exc, ty2) = check_expr m.matrixdecl_cols env in
405         env.scope.vars <- (m.matrixdecl_type, m.matrixdecl_name,
406             C_Matrix_Index(m.matrixdecl_name, exr, exc, m.matrixdecl_type))::env.
                  scope.vars;
407         let typ = m.matrixdecl_type in
408         let errorstring = "Error with " in
409         let rows = string_of_expr m.matrixdecl_rows in
410         let cols = string_of_expr m.matrixdecl_cols in
411         try
412         if int_of_string rows = List.length el && int_of_string cols > -1 then
413           (* Inside here need to call my functions from above for matrix stuff *)
414           let ret = check_matrix_vals m el (int_of_string cols) typ env in
415             if ret = m then
416               C_MatrixInit( {Stch_cast.matrixdecl_name = m.matrixdecl_name;
417                 Stch_cast.matrixdecl_type = m.matrixdecl_type;
418                 Stch_cast.matrixdecl_rows = m.matrixdecl_rows;
419                 Stch_cast.matrixdecl_cols = m.matrixdecl_cols}, el)
420             else begin
421               (* print_string "HELLO"; *)
422               raise(Error(errorstring ^ "checking return value of list iter"))
423             end
424         else begin
425           (* print_string "HELLO2"; *)
426           raise(Error(errorstring ^ "Int of string statement failure"))
```

```
427         end
428         with
429         | _ -> begin
430           (* print_string "HELLO3"; *)
431           raise(Error(errorstring ^ "try/with failure"))
432           end
433       end
434
435   and check_matrix_decl (m: matrixdecl) (env: stch_env) =
436
437     (* create a variable declaration out of the array declaration so we can check
          for it *)
438     let mat = { Stch_ast.vdecl_type = m.matrixdecl_type;
439           Stch_ast.vdecl_name = m.matrixdecl_name} in
440
441
442     (* check to see if the variable is not already declared *)
443     let invalid = List.exists (fun (_, s, _) -> s = mat.vdecl_name) env.scope.vars
          in
444     if invalid then
445       raise (Error("Variable " ^ mat.vdecl_name ^ " already declared"))
446     else
447     (* if it isn't, put it in the scope, and make a new c_arraydecl
448       after you typematch the size expression *)
449       let (exr, ty) = check_expr m.matrixdecl_rows env in
450       let (exc, ty) = check_expr m.matrixdecl_cols env in
451       env.scope.vars <- (mat.vdecl_type, mat.vdecl_name,
452         C_Matrix_Index(mat.vdecl_name, exr, exc ,mat.vdecl_type))::env.scope.vars;
453       let (row, typerow) = check_expr m.matrixdecl_rows env in
454       let (col, typecol) = check_expr m.matrixdecl_cols env in
455       match (typerow, typecol) with
456         (Tfloat, _) -> raise (Error("Invalid matrix row type, expects int"))
457         | (Tchar, _) -> raise (Error("Invalid matrix row type, expects int"))
458         | (Tstring, _) -> raise (Error("Invalid matrix row type, expects int"))
459         | (_, Tstring) -> raise (Error("Invalid matrix col type, expects int"))
460         | (_, Tfloat) -> raise (Error("Invalid matrix col type, expects int"))
461         | (_, Tchar) -> raise (Error("Invalid matrix col type, expects int"))
462         | (Tint, Tvoid) -> raise (Error("Invalid matrix row type, expects int"))
463         | (Tvoid, Tint) -> raise (Error("Invalid matrix row type, expects int"))
464         | (Tvoid, Tvoid) -> raise (Error("Invalid matrix decl. Must be 2 ints"))
465       (* else it's a void or an int, and it's allowed *)
466         | _ -> let v = { Stch_cast.matrixdecl_type = mat.vdecl_type;
467                 Stch_cast.matrixdecl_name = mat.vdecl_name;
468                 Stch_cast.matrixdecl_rows = m.matrixdecl_rows;
469                 Stch_cast.matrixdecl_cols = m.matrixdecl_cols} in
470                   C_MatrixDecl(v)
471
472
473   (* Typechecking the expression of an "if" statement *)
474   and check_if (ex: expr) (th: stmt) (el: stmt) (en : stch_env) =
475     let (e, t) = check_expr ex en in
476       if t = Tint || t = Tfloat || t = Tchar then
477         let s1 = check_stmt th en in
478         let s2 = check_stmt el en in
479         C_If(e, s1, s2)
480       else
481         raise (Error("If clause has expression of type " ^ string_of_dataType t))
482
```

```
483
484   (* typecheck the for loop *)
485   and check_for (e1: expr) (e2: expr) (e3: expr) (st: stmt) (env: stch_env) =
486     let (ex1, t1) = check_expr e1 env in
487     let (ex2, t2) = check_expr e2 env in
488     let (ex3, t3) = check_expr e3 env in
489     if t1 <> Tint && t1 <> Tvoid then
490       raise (Error("For Loop: First expression not of type int."))
491     else begin
492       if t2 <> Tint && t2 <> Tvoid then
493         raise (Error("For Loop: Second expression not of type int."))
494       else begin
495         if t3 <> Tint && t3 <> Tvoid then
496           raise (Error("For Loop: Third expression not of type int."))
497         else begin
498           let s = check_stmt st env in
499           C_For(ex1,ex2,ex3,s)
500         end
501       end
502     end
503
504
505
506 (* Go through the body of a stitch loop and create an environment of all the
      variables used, so we know
507   what needs to be passed in
508   NOTE: VDECLS and ARRAYDECLS/MATRIXDECLS should NOT be added here, because those
      are local in the stitch
509   loop and should not be copied *)
510
511   and check_stitch_body (el: c_stmt list) (table: symTable) (env: stch_env) = match
      el with
512   [] -> table
513   | head::tail ->
514   (match head with
515     (* The symtable of block here consists of all the variables that I do not want
        to put in the struct,
516       so we just pass the list through *)
517     | C_Block(t, b) -> check_stitch_body b table env
518     | C_Vdecl(a) -> let n = a.vdecl_name in
519       let table' = {Stch_cast.parent = table.parent; Stch_cast.vars =
520       List.filter ( fun (typ,nm,ex) -> nm <> n ) env.scope.vars } in
521       check_stitch_body tail table' env
522     | C_ArrayDecl(a) -> let n = a.arraydecl_name in
523       let table' = {Stch_cast.parent = table.parent; Stch_cast.vars =
524       List.filter ( fun (typ,nm,ex) -> nm <> n ) env.scope.vars } in
525       check_stitch_body tail table' env
526     | C_MatrixDecl(m) -> let n = m.matrixdecl_name in
527       let table' = {Stch_cast.parent = table.parent; Stch_cast.vars =
528       List.filter ( fun (typ,nm,ex) -> nm <> n ) env.scope.vars } in
529       check_stitch_body tail table' env
530     | C_Assign(v, r) -> let n = v.vdecl_name in
531       let table' = {Stch_cast.parent = table.parent; Stch_cast.vars =
532       List.filter ( fun (typ,nm,ex) -> nm <> n ) env.scope.vars } in
533       check_stitch_body tail table' env
534
535     | C_ArrayInit(a, el) -> let n = a.arraydecl_name in
536       let table' = {Stch_cast.parent = table.parent; Stch_cast.vars =
```

```
537       List.filter ( fun (typ,nm,ex) -> nm <> n ) env.scope.vars } in
538       check_stitch_body tail table' env
539     | C_MatrixInit(m, el) -> let n = m.matrixdecl_name in
540       let table' = {Stch_cast.parent = table.parent; Stch_cast.vars =
541       List.filter ( fun (typ,nm,ex) -> nm <> n ) env.scope.vars } in
542       check_stitch_body tail table' env
543
544     (* else I want to keep them in the symtable, continue down the list *)
545     | _ -> check_stitch_body tail table env
546   )
547
548
549   (* Iterate through all the variables, adding them to one symtable *)
550   and iterate_vars (data: (dataType * string * c_expr) list ) (table: symTable) =
        match data with
551     | [] -> table
552     | head::tail ->  ignore(table.vars <- head::table.vars);
553             iterate_vars tail table
554
555   (* Bounce up each symtable level, constructing one large symtable with all the
        variables *)
556     and check_all_envs (el: c_stmt list) (currTable: symTable) (newTable: symTable)
          (env: stch_env) =
557
558       ignore(iterate_vars currTable.vars newTable); (* add all the vars to the
            current table *)
559     match currTable.parent with (* then check the parent *)
560       | None -> newTable
561         | Some(parent) -> check_all_envs el parent newTable env
562
563
564
565   (* Typechecking the expressions of a Stitch Loop *)
566   and check_stitch (var : expr) (start : expr) (s_end : expr) (stride : expr) (body
        : stmt) (env : stch_env) =
567     let (var', t1) = check_expr var env in
568     let name = get_id_from_expr var in
569     let (start', t2) = check_expr start env in
570     let (s_end', t3) = check_expr s_end env in
571     let (stride', t4) = check_expr stride env in
572     if t1 <> Tint then raise (Error("Stitch: First expression not of type int."))
573     else begin
574       if t2 <> Tint then raise (Error("Stitch: Second expression not of type int."))
575       else begin
576         if t3 <> Tint then raise (Error("Stitch: Third expression not of type int.")
              )
577         else begin
578           if t4 <> Tint then raise (Error("Stitch: Fourth expression not of type int
                "))
579           else begin
580             let body' = [(check_stmt body env)] in
581             let n' = check_all_envs body' env.scope {Stch_cast.parent = None;
                  Stch_cast.vars = []} env in
582             let t' = check_stitch_body body' n' env in
583             let scope' = {Stch_cast.parent = env.scope.parent;
584               Stch_cast.vars = List.filter (fun (t, n, e) -> n <> name) t'.vars } in
585               C_Stitch(var', start', s_end', stride', gen_name sn, body', scope')
586           end
```

```
587         end
588       end
589     end
590
591   (* typecheck the while loop *)
592   and check_while (e: expr) (s: stmt) (env: stch_env) =
593     let (e,t) = check_expr e env in
594     if t = Tint then
595       let s' = check_stmt s env in C_While(e,s')
596     else
597       raise (Error("Invalid 'while' expression"))
598
599 let check_formals (decl: vdecl) (env: stch_env) =
600   match decl.vdecl_type with
601     dataType -> env.scope.vars <- (decl.vdecl_type, decl.vdecl_name, C_Noexpr)::env.
          scope.vars;
602       let v = { Stch_cast.vdecl_type = decl.vdecl_type;
603             Stch_cast.vdecl_name = decl.vdecl_name } in v
604
605 let check_for_ret (body: stmt list) =
606   if (List.exists ( fun ( s ) -> match s with
607                       Return(a) -> true
608                       | _ -> false ) body) then ""
609   else
610     raise (Error("Control reaches the end of nonvoid function."))
611
612 (* typecheck a function declaration *)
613 let check_fdecl (func: Stch_ast.fdecl) (env: stch_env) : c_fdecl =
614   if env.in_func then
615     raise (Error ("Cannot declare a function within another function"))
616   else
617     let env' = { env with scope = {parent = Some(env.scope); vars = [];};
618     retType = func.fdecl_type; in_func = true} in
619     let f_formals = (List.rev (List.map (fun x -> check_formals x env') func.
          fdecl_formals)) in
620       let f = { Stch_cast.fdecl_name = func.fdecl_name;
621             Stch_cast.fdecl_type = func.fdecl_type;
622             Stch_cast.fdecl_formals = f_formals;
623             Stch_cast.body = ( List.map (fun x -> check_stmt x env') func.body );}
                  in
624               match func.fdecl_type with
625                 Tvoid -> env.funcs <- f::env.funcs; f
626                 | _ -> ignore(check_for_ret func.body); env.funcs <- f::env.funcs; f
627
628
629 (* typecheck the ast env *)
630 let init_env : (stch_env) =
631   let init_funcs = [{ fdecl_type = Tvoid;
632             fdecl_name = "print";
633             fdecl_formals = [ {vdecl_type = Tstring; vdecl_name = "c"}; ];
634             body = [];
635             };
636
637             {fdecl_type = Tvoid;
638              fdecl_name = "error";
639              fdecl_formals = [ {vdecl_type = Tstring; vdecl_name = "c"}; ];
640              body = [];
641             };
```

```
642
643             {fdecl_type = Tvoid;
644              fdecl_name = "exit";
645              fdecl_formals = [ {vdecl_type = Tint; vdecl_name = "c"}; ];
646              body = [];
647             };
648
649             {fdecl_type = Tfile;
650              fdecl_name = "open_r";
651              fdecl_formals = [ {vdecl_type = Tstring; vdecl_name = "fn"}; ];
652              body = [];
653             };
654
655             {fdecl_type = Tfile;
656              fdecl_name = "open_w";
657              fdecl_formals = [ {vdecl_type = Tstring; vdecl_name = "fn"}; ];
658              body = [];
659             };
660
661             {fdecl_type = Tint;
662              fdecl_name = "read";
663              fdecl_formals = [ {vdecl_type = Tfile; vdecl_name = "f"}; {vdecl_type =
                     Tchar; vdecl_name = "a"}; ];
664              body = [];
665             };
666
667             {fdecl_type = Tint;
668              fdecl_name = "write";
669              fdecl_formals = [ {vdecl_type = Tfile; vdecl_name = "f"}; {vdecl_type =
                     Tchar; vdecl_name = "a"}; ];
670              body = [];
671             };
672             ] in (* Need to add builtin functions here *)
673   let init_scope = { parent = None; vars = []; } in
674   { funcs = init_funcs;
675     scope = init_scope;
676     retType = Tvoid;
677     in_func = false;
678   }
679
680
681 (* check the programc *)
682 let check_prog (prog: Stch_ast.program) : (Stch_cast.c_program) =
683   let env = init_env in
684 { Stch_cast.stmts = (List.map (fun x -> check_stmt x env) (fst prog));
685   Stch_cast.funcs = (List.map (fun x -> check_fdecl x env) (List.rev (snd prog)));
686   Stch_cast.syms = env.scope;
687 }
```

stch_cast.ml

```
1 (*
2 C AST
3 December 2015
4 Authors: Dan Cole & Tim Waterman
5
6 The C AST that will be generated from our semantic analysis
7 *)
8
9 open Stch_ast
10
11 (* Expressions *)
12 type c_expr =
13     C_Int of int
14   | C_Float of float
15   | C_Char of char
16   | C_Escape of string
17   | C_Id of string * dataType
18   | C_String of string
19   | C_Binop of c_expr * op * c_expr
20   | C_Negate of c_expr
21   | C_Call of string * c_expr list
22   | C_Assign2 of string * c_expr
23   | C_Array_Index of string * c_expr * dataType
24   | C_Matrix_Index of string * c_expr * c_expr * dataType
25   | C_Array_Item_Assign of string * c_expr * c_expr
26   | C_Matrix_Item_Assign of string * c_expr * c_expr * c_expr
27   | C_Noexpr
28
29 (* Symbol table to store variable and function names *)
30 type symTable = {
31   parent: symTable option;
32   mutable vars: (dataType * string * c_expr) list;
33 }
34
35 type c_vdecl = {
36   vdecl_type : dataType;
37   vdecl_name : string;
38 }
39
40 (* Array and Matrix data types *)
41 type c_arraydecl = {
42   arraydecl_type : dataType;
43   arraydecl_name : string;
44   arraydecl_size : expr;
45 }
46
47 type c_matrixdecl = {
48   matrixdecl_type : dataType;
49   matrixdecl_name : string;
50   matrixdecl_rows : expr;
51   matrixdecl_cols : expr;
52   }
53
54 (* Statements *)
```

```
55 type c_stmt =
56     C_Block of symTable * c_stmt list
57   | C_Vdecl of c_vdecl
58   | C_ArrayDecl of c_arraydecl
59   | C_ArrayInit of c_arraydecl * expr list
60   | C_MatrixInit of c_matrixdecl * expr list list
61   | C_MatrixDecl of c_matrixdecl
62   | C_Expr of dataType * c_expr
63   | C_Return of dataType * c_expr
64   | C_If of c_expr * c_stmt * c_stmt
65   | C_For of c_expr * c_expr * c_expr * c_stmt
66   | C_While of c_expr * c_stmt
67   | C_Stitch of c_expr * c_expr * c_expr * c_expr * string * c_stmt list * symTable
68   | C_Assign of c_vdecl * c_expr
69   | C_Break
70
71 type c_fdecl = {
72     fdecl_type     : dataType;
73     fdecl_name     : string;
74     fdecl_formals    : c_vdecl list;
75     body       : c_stmt list;
76   }
77
78 (* Our environment *)
79 type stch_env = {
80   mutable funcs: c_fdecl list;
81   scope: symTable;
82   retType: dataType;
83   in_func: bool;
84 }
85
86 type c_program = {
87   stmts : c_stmt list;
88   funcs : c_fdecl list;
89   syms  : symTable;
90 }
```

c_generator.ml

```
1 (*
2 C Code Generator
3 December 2015
4 Authors: Dan Cole & Tim Waterman
5
6 Takes the C_AST and Generates the corresponding C Code
7 *)
8
9 open Stch_ast
10 open Stch_cast
11 exception Error of string
12
13 let string_of_c_dataType = function
14     Tint -> "int"
15   | Tfloat -> "float"
16   | Tchar -> "char"
17   | Tvoid -> "void"
18   | Tstring -> "char *"
19   | Tintap -> "int"
20   | Tintam -> "int"
21   | Tfloatap -> "float"
22   | Tfloatam -> "float"
23   | Tfile -> "FILE *"
24
25 (* Generates the c code for the corresponding expression from our C_AST *)
26 let rec string_of_c_expr = function
27     C_Int(l) -> string_of_int l
28   | C_Float(l) -> string_of_float l
29   | C_Char(l) ->  "\'" ^ String.make 1 l ^ "\'"
30   | C_Escape(l) -> "\'" ^ l ^ "\'"
31   | C_Id(s, t) -> s
32   | C_String(s) -> "\"" ^ s ^ "\""
33   | C_Binop(e1, o, e2) ->
34       "(" ^ string_of_c_expr e1 ^ " " ^
35       (match o with
36         Add -> "+" | Subtract -> "-" | Times -> "*" | Divide -> "/"
37       | Equal -> "==" | Ne -> "!="
38       | Lt -> "<" | Le -> "<=" | Gt -> ">" | Ge -> ">="
39       | Or -> "||" | And -> "&&" | Mod -> "%" ) ^ " " ^
40       string_of_c_expr e2 ^ ")"
41   | C_Negate(e) -> "!" ^ string_of_c_expr e
42
43   (* For call, we need to match our various built-in functions *)
44   | C_Call(f, el) -> (match f with "print" -> "printf"
45                                  | "error" -> "fprintf"
46                                  | "open_r" -> "fopen"
47                                  | "open_w" -> "fopen"
48                                  | "read" -> "fread"
49                                  | "write" -> "fwrite"
50                                  | _ -> f) ^
51                      "(" ^ String.concat ", " (match f with "print" -> print_2_fprint
                            (List.hd el)
52                                                          | "error" ->
                                                              error_2_fprintf (List.
```

```
                                                          hd el)
53                                                      | "open_r" ->
                                                          open_2_fopen_r (List.
                                                          hd el)
54                                                      | "open_w" ->
                                                          open_2_fopen_w (List.
                                                          hd el)
55                                                      | "read" -> read_2_fread
                                                          el
56                                                      | "write" ->
                                                          write_2_fwrite el
57                                                      | _ -> List.map
                                                          string_of_c_expr el) ^
                                                           ")"
58    | C_Assign2(i, e) -> i ^ " = " ^ string_of_c_expr e
59    | C_Array_Item_Assign(id, ind, e) -> id ^ "[" ^ string_of_c_expr ind ^"] = " ^
        string_of_c_expr e
60    | C_Array_Index(a, i, t) -> a ^ "[" ^ string_of_c_expr i ^ "]"
61    | C_Matrix_Index(m, r, c, t) -> m ^ "[" ^ string_of_c_expr r ^ "][" ^
        string_of_c_expr c ^ "]"
62    | C_Matrix_Item_Assign(m, r, c, e) -> m ^ "[" ^ string_of_c_expr r ^ "][" ^
        string_of_c_expr c ^ "] = " ^ string_of_c_expr e
63    | C_Noexpr -> ""
64
65      (* Converting from read to the C function fread() *)
66      and read_2_fread (el: c_expr list) =
67        let file = List.hd el in
68          let arr = List.hd (List.rev el) in
69            match file with
70              C_Id(s, t) -> (match t with
71                Tfile -> (match arr with
72                  C_Id(s', t') -> (s' ^ ", sizeof(" ^ s' ^ "), sizeof(" ^
                      string_of_c_dataType t' ^ "), " ^ s)::[]
73                  | _ -> raise(Error("Invalid argument type for read: " ^
                      string_of_c_expr arr)))
74                | _ -> raise(Error("Invalid argument type for read: " ^
                    string_of_c_expr file)))
75              | _ -> raise(Error("Invalid argument for read: " ^ string_of_c_expr
                    file))
76
77      (* Converting for write to the C function fwrite() *)
78      and write_2_fwrite (el: c_expr list) =
79        let file = List.hd el in
80          let arr = List.hd (List.rev el) in
81            match file with
82              C_Id(s, t) -> (match t with
83                Tfile -> (match arr with
84                  C_Id(s', t') -> (s' ^ ", sizeof(" ^ s' ^ "), sizeof(" ^
                      string_of_c_dataType t' ^ "), " ^ s)::[]
85                  | _ -> raise(Error("Invalid argument type for read: " ^
                      string_of_c_expr arr)))
86                | _ -> raise(Error("Invalid argument type for read: " ^
                    string_of_c_expr file)))
87              | _ -> raise(Error("Invalid argument for read: " ^ string_of_c_expr
                    file))
88
89      (* Converting the two open functions *)
90      and open_2_fopen_r (e: c_expr) = match e with
```

```
91        C_String(l) -> ("\"" ^ l ^ "\", \"r+\"" )::[]
92      | _ -> raise (Error("Invalid argument for open: " ^ string_of_c_expr e))
93
94      and open_2_fopen_w (e: c_expr) = match e with
95        C_String(l) -> ("\"" ^ l ^ "\", \"w+\"" )::[]
96      | _ -> raise (Error("Invalid argument for open: " ^ string_of_c_expr e))
97
98    (* Generating print statements based on args *)
99      and print_2_fprint (e: c_expr) = match e with
100       C_Int(l) -> ("\"%d\\n\", " ^ string_of_c_expr e)::[]
101     | C_Float(l) -> ("\"%f\\n\", " ^ string_of_c_expr e)::[]
102     | C_Char(l) -> ("\"%c\\n\", " ^ string_of_c_expr e)::[]
103     | C_String(l) -> ("\"%s\\n\", " ^ string_of_c_expr e)::[]
104     | C_Array_Index(a, i, t) -> (match t with
105                                    Tint | Tintap | Tintam -> ("\"%d\\n\", " ^ a ^
                                         "[" ^ string_of_c_expr i ^ "]")::[]
106                                  | Tfloat | Tfloatap | Tfloatam -> ("\"%f\\n\", "
                                       ^ a ^ "[" ^ string_of_c_expr i ^ "]")::[]
107                                  | Tchar -> ("\"%c\\n\", " ^ a ^ "[" ^
                                       string_of_c_expr i ^ "]")::[]
108                                  | Tstring -> ("\"%s\\n\", " ^ a ^ "[" ^
                                       string_of_c_expr i ^ "]")::[]
109                                  | Tvoid -> raise (Error("Invalid print type Void
                                       : " ^ a ^ "[" ^ string_of_c_expr i ^ "]"))
110                                  | Tfile -> raise (Error("Invalid print type File
                                       ")))
111     | C_Matrix_Index(m, r, c, t) -> (match t with
112                                     Tint | Tintap | Tintam -> ("\"%d\\n\", " ^ m ^
                                          "[" ^ string_of_c_expr r ^ "][" ^
113                                               string_of_c_expr c ^ "]")::[]
114                                  | Tfloat | Tfloatap | Tfloatam -> ("\"%f\\n\", "
                                       ^ m ^ "[" ^ string_of_c_expr r ^ "][" ^
115                                               string_of_c_expr c ^ "]")::[]
116                                  | Tchar -> ("\"%c\\n\", " ^ m ^ "[" ^
                                       string_of_c_expr r ^ "][" ^
117                                               string_of_c_expr c ^ "]")::[]
118                                  | Tstring -> ("\"%s\\n\", " ^ m ^ "[" ^
                                       string_of_c_expr r ^ "][" ^
119                                               string_of_c_expr c ^ "]")::[]
120                                  | Tvoid -> raise(Error("Invalid print type void
                                       in matrix printing"))
121                                  | Tfile -> raise(Error("Invlaid print type file
                                       in matrix printing")))
122     | C_Id(l, t) -> (match t with
123                       Tint | Tintap | Tintam -> ("\"%d\\n\", " ^ string_of_c_expr
                            e)::[]
124                     | Tfloat | Tfloatap | Tfloatam -> ("\"%f\\n\", " ^
                            string_of_c_expr e)::[]
125                     | Tchar -> ("\"%c\\n\", " ^ string_of_c_expr e)::[]
126                     | Tstring -> ("\"%s\\n\", " ^ string_of_c_expr e)::[]
127                     | Tvoid -> raise (Error("Invalid print type Void: " ^
                            string_of_c_expr e))
128                     | Tfile -> raise (Error("Invalid print type File: ")))
129     | C_Binop(lhs, o, rhs) -> (match o with
130                                  Add -> (match lhs with
131                                            C_Int(l) -> ("\"%d\\n\", " ^
                                                 string_of_c_expr lhs ^ "+" ^
                                                 string_of_c_expr rhs)::[]
```

```
132                                     | C_Float(l) -> ("\"%f\\n\", " ^
                                          string_of_c_expr lhs ^ "+" ^
                                          string_of_c_expr rhs)::[]
133                                     | C_Id(l, t) -> (match t with
134                                                       Tint | Tintap |
                                                          Tintam ->
                                                          ("\"%d\\n\", "
                                                           ^
                                                          string_of_c_expr
                                                           lhs ^ "+" ^
                                                          string_of_c_expr
                                                           rhs)::[]
135                                                     | Tfloat | Tfloatap
                                                          | Tfloatam ->
                                                          ("\"%f\\n\", " ^

                                                          string_of_c_expr
                                                           lhs ^ "+" ^
                                                          string_of_c_expr
                                                           rhs)::[]
136                                                     | Tchar -> ("\"%c\\n
                                                          \", " ^
                                                          string_of_c_expr
                                                           lhs ^ "+" ^
                                                          string_of_c_expr
                                                           rhs)::[]
137                                                     | Tstring -> ("\"%s
                                                          \\n\", " ^
                                                          string_of_c_expr
                                                           lhs ^ "+" ^
                                                          string_of_c_expr
                                                           rhs)::[]
138                                                     | Tvoid -> raise (
                                                          Error("Invalid
                                                          print type Void:
                                                           " ^
                                                          string_of_c_expr
                                                           lhs ^ "+" ^
                                                          string_of_c_expr
                                                           rhs))
139                                                     | Tfile -> raise (
                                                          Error("Invalid
                                                          print type File:
                                                           ")))
140                                     | _ -> raise (Error("Invalid add in
                                          function call"))
141                                 )
142                               | Subtract -> (match lhs with
143                                              C_Int(l) -> ("\"%d\\n\", " ^
                                                 string_of_c_expr lhs ^ "-" ^
                                                 string_of_c_expr rhs)::[]
144                                             | C_Float(l) -> ("\"%f\\n\", " ^
                                                string_of_c_expr lhs ^ "-" ^
                                                string_of_c_expr rhs)::[]
145                                             | C_Id(l, t) -> (match t with
146                                                               Tint | Tintap |
                                                                  Tintam ->
                                                                  ("\"%d\\n\", "
```

```
                                                                ^
                                                                  string_of_c_expr
                                                                   lhs ^ "-" ^
                                                                  string_of_c_expr
                                                                   rhs)::[]
147                                                           | Tfloat | Tfloatap
                                                                | Tfloatam ->
                                                                ("\"%f\\n\", " ^

                                                                  string_of_c_expr
                                                                   lhs ^ "-" ^
                                                                  string_of_c_expr
                                                                   rhs)::[]
148                                                           | Tchar -> ("\"%c\\n
                                                                \", " ^
                                                                  string_of_c_expr
                                                                   lhs ^ "-" ^
                                                                  string_of_c_expr
                                                                   rhs)::[]
149                                                           | Tstring -> ("\"%s
                                                                \\n\", " ^
                                                                  string_of_c_expr
                                                                   lhs ^ "-" ^
                                                                  string_of_c_expr
                                                                   rhs)::[]
150                                                           | Tvoid -> raise (
                                                                Error("Invalid
                                                                print type Void:
                                                                " ^
                                                                  string_of_c_expr
                                                                   lhs ^ "-" ^
                                                                  string_of_c_expr
                                                                   rhs))
151                                                           | Tfile -> raise (
                                                                Error("Invalid
                                                                print type File:
                                                                ")))
152                                                 | _ -> raise (Error("Invalid add in
                                                        function call"))
153                                             )
154                                       | Times -> (match lhs with
155                                                   C_Int(l) -> ("\"%d\\n\", " ^
                                                       string_of_c_expr lhs ^ "*" ^
                                                       string_of_c_expr rhs)::[]
156                                                 | C_Float(l) -> ("\"%f\\n\", " ^
                                                      string_of_c_expr lhs ^ "*" ^
                                                      string_of_c_expr rhs)::[]
157                                                 | C_Id(l, t) -> (match t with
158                                                                    Tint | Tintap |
                                                                        Tintam ->
                                                                        ("\"%d\\n\", "
                                                                         ^
                                                                        string_of_c_expr
                                                                         lhs ^ "*" ^
                                                                        string_of_c_expr
                                                                         rhs)::[]
159                                                                 | Tfloat | Tfloatap
                                                                      | Tfloatam ->
```

```
                                                    ("\"%f\\n\", " ^
                                                     string_of_c_expr
                                                      lhs ^ "*" ^
                                                     string_of_c_expr
                                                      rhs)::[]
160                                                | Tchar -> ("\"%c\\n
                                                     \", " ^
                                                     string_of_c_expr
                                                      lhs ^ "*" ^
                                                     string_of_c_expr
                                                      rhs)::[]
161                                                | Tstring -> ("\"%s
                                                     \\n\", " ^
                                                     string_of_c_expr
                                                      lhs ^ "*" ^
                                                     string_of_c_expr
                                                      rhs)::[]
162                                                | Tvoid -> raise (
                                                     Error("Invalid
                                                     print type Void:
                                                      " ^
                                                     string_of_c_expr
                                                      lhs ^ "*" ^
                                                     string_of_c_expr
                                                      rhs))
163                                                | Tfile -> raise (
                                                     Error("Invalid
                                                     print type File:
                                                      ")))
164                                  | _ -> raise (Error("Invalid add in
                                        function call"))
165                                )
166                              | Divide -> (match lhs with
167                                  C_Int(l) -> ("\"%d\\n\", " ^
                                      string_of_c_expr lhs ^ "/" ^
                                      string_of_c_expr rhs)::[]
168                                | C_Float(l) -> ("\"%f\\n\", " ^
                                    string_of_c_expr lhs ^ "/" ^
                                    string_of_c_expr rhs)::[]
169                                | C_Id(l, t) -> (match t with
170                                                   Tint | Tintap |
                                                     Tintam ->
                                                     ("\"%d\\n\", "
                                                      ^
                                                     string_of_c_expr
                                                      lhs ^ "/" ^
                                                     string_of_c_expr
                                                      rhs)::[]
171                                                | Tfloat | Tfloatap
                                                     | Tfloatam ->
                                                    ("\"%f\\n\", " ^
                                                     string_of_c_expr
                                                      lhs ^ "/" ^
                                                     string_of_c_expr
                                                      rhs)::[]
172                                                | Tchar -> ("\"%c\\n
```

```
                                                                \", " ^
                                                                string_of_c_expr
                                                                 lhs ^ "/" ^
                                                                string_of_c_expr
                                                                 rhs)::[]
173                                                           | Tstring -> ("\"%s
                                                                \\n\", " ^
                                                                string_of_c_expr
                                                                 lhs ^ "/" ^
                                                                string_of_c_expr
                                                                 rhs)::[]
174                                                           | Tvoid -> raise (
                                                                Error("Invalid
                                                                print type Void:
                                                                " ^
                                                                string_of_c_expr
                                                                 lhs ^ "/" ^
                                                                string_of_c_expr
                                                                 rhs))
175                                                           | Tfile -> raise (
                                                                Error("Invalid
                                                                print type File:
                                                                ")))
176                                          | _ -> raise (Error("Invalid add in
                                                 function call"))
177                                    )
178                                | Equal -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                     "==" ^ string_of_c_expr rhs)::[]
179                                | Ne -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                     "!=" ^ string_of_c_expr rhs)::[]
180                                | Lt -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                     "<" ^ string_of_c_expr rhs)::[]
181                                | Le -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                     "<=" ^ string_of_c_expr rhs)::[]
182                                | Gt -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                     ">" ^ string_of_c_expr rhs)::[]
183                                | Ge -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                     ">=" ^ string_of_c_expr rhs)::[]
184                                | Or -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                     "||" ^ string_of_c_expr rhs)::[]
185                                | And -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                     "&&" ^ string_of_c_expr rhs)::[]
186                                | Mod -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                     "%" ^ string_of_c_expr rhs)::[]
187                                    )
188       | _ -> raise (Error("Invalid expr in print statement: " ^ string_of_c_expr e))
189
190     (* Generating the error function based on parameters *)
191       and error_2_fprintf (e: c_expr) = match e with
192         C_Int(l) -> ("stderr, \"%d\\n\", " ^ string_of_c_expr e)::[]
193       | C_Float(l) -> ("stderr, \"%f\\n\", " ^ string_of_c_expr e)::[]
194       | C_Char(l) -> ("stderr, \"%c\\n\", " ^ string_of_c_expr e)::[]
195       | C_String(l) -> ("stderr, \"%s\\n\", " ^ string_of_c_expr e)::[]
196       | C_Id(l, t) -> (match t with
197                          Tint | Tintap | Tintam -> ("stderr, \"%d\\n\", " ^
                               string_of_c_expr e)::[]
198                        | Tfloat | Tfloatap | Tfloatam -> ("stderr, \"%f\\n\", " ^
                               string_of_c_expr e)::[]
```

```
199                         | Tchar -> ("stderr, \"%c\\n\", " ^ string_of_c_expr e)::[]
200                         | Tstring -> ("stderr, \"%s\\n\", " ^ string_of_c_expr e)
                              ::[]
201                         | Tvoid -> raise (Error("Invalid print type Void: " ^
                              string_of_c_expr e))
202                         | Tfile -> raise (Error("Invalid print type File: ")))
203       | C_Binop(lhs, o, rhs) -> (match o with
204                                    Add -> (match lhs with
205                                              C_Int(l) -> ("stderr, \"%d\\n\", " ^
                                                  string_of_c_expr lhs ^ "+" ^
                                                  string_of_c_expr rhs)::[]
206                                            | C_Float(l) -> ("stderr, \"%f\\n\", " ^
                                                string_of_c_expr lhs ^ "+" ^
                                               string_of_c_expr rhs)::[]
207                                            | C_Id(l, t) -> (match t with
208                                                                Tint | Tintap |
                                                                    Tintam -> ("
                                                                    stderr, \"%d\\
                                                                    n\", " ^
                                                                    string_of_c_expr
                                                                     lhs ^ "+" ^
                                                                    string_of_c_expr
                                                                     rhs)::[]
209                                                               | Tfloat | Tfloatap
                                                                  | Tfloatam -> ("
                                                                  stderr, \"%f\\n
                                                                  \", " ^
                                                                  string_of_c_expr
                                                                   lhs ^ "+" ^
                                                                  string_of_c_expr
                                                                   rhs)::[]
210                                                               | Tchar -> ("stderr,
                                                                   \"%c\\n\", " ^
                                                                  string_of_c_expr
                                                                   lhs ^ "+" ^
                                                                  string_of_c_expr
                                                                   rhs)::[]
211                                                               | Tstring -> ("
                                                                  stderr, \"%s\\n
                                                                  \", " ^
                                                                  string_of_c_expr
                                                                   lhs ^ "+" ^
                                                                  string_of_c_expr
                                                                   rhs)::[]
212                                                               | Tvoid -> raise (
                                                                  Error("Invalid
                                                                  print type Void:
                                                                   " ^
                                                                  string_of_c_expr
                                                                   lhs ^ "+" ^
                                                                  string_of_c_expr
                                                                   rhs))
213                                                               | Tfile -> raise (
                                                                  Error("Invalid
                                                                  print type File:
                                                                   ")))
214                                            | _ -> raise (Error("Invalid add in
                                                  function call")))
```

```
215                                     )
216                                 | Subtract -> (match lhs with
217                                     C_Int(l) -> ("stderr, \"%d\\n\", " ^
                                         string_of_c_expr lhs ^ "-" ^
                                         string_of_c_expr rhs)::[]
218                                   | C_Float(l) -> ("stderr, \"%f\\n\", " ^
                                       string_of_c_expr lhs ^ "-" ^
                                      string_of_c_expr rhs)::[]
219                                   | C_Id(l, t) -> (match t with
220                                                      Tint | Tintap |
                                                          Tintam -> ("
                                                          stderr, \"%d\\
                                                          n\", " ^
                                                          string_of_c_expr
                                                           lhs ^ "-" ^
                                                          string_of_c_expr
                                                           rhs)::[]
221                                                    | Tfloat | Tfloatap
                                                        | Tfloatam -> ("
                                                        stderr, \"%f\\n
                                                        \", " ^
                                                        string_of_c_expr
                                                         lhs ^ "-" ^
                                                        string_of_c_expr
                                                         rhs)::[]
222                                                    | Tchar -> ("stderr,
                                                         \"%c\\n\", " ^
                                                        string_of_c_expr
                                                         lhs ^ "-" ^
                                                        string_of_c_expr
                                                         rhs)::[]
223                                                    | Tstring -> ("
                                                        stderr, \"%s\\n
                                                        \", " ^
                                                        string_of_c_expr
                                                         lhs ^ "-" ^
                                                        string_of_c_expr
                                                         rhs)::[]
224                                                    | Tvoid -> raise (
                                                        Error("Invalid
                                                        print type Void:
                                                         " ^
                                                        string_of_c_expr
                                                         lhs ^ "-" ^
                                                        string_of_c_expr
                                                         rhs))
225                                                    | Tfile -> raise (
                                                        Error("Invalid
                                                        print type File:
                                                         ")))
226                                   | _ -> raise (Error("Invalid add in
                                       function call"))
227                                     )
228                                 | Times -> (match lhs with
229                                     C_Int(l) -> ("stderr, \"%d\\n\", " ^
                                         string_of_c_expr lhs ^ "*" ^
                                         string_of_c_expr rhs)::[]
230                                   | C_Float(l) -> ("stderr, stderr, \"%f\\
```

```
                                                n\", " ^ string_of_c_expr lhs ^ "*"
                                                ^ string_of_c_expr rhs)::[]
231                                         | C_Id(l, t) -> (match t with
232                                                           Tint | Tintap |
                                                              Tintam -> ("
                                                              stderr, \"%d\\
                                                              n\", " ^
                                                              string_of_c_expr
                                                               lhs ^ "*" ^
                                                              string_of_c_expr
                                                               rhs)::[]
233                                                         | Tfloat | Tfloatap
                                                            | Tfloatam -> ("
                                                            stderr, \"%f\\n
                                                            \", " ^
                                                            string_of_c_expr
                                                             lhs ^ "*" ^
                                                            string_of_c_expr
                                                             rhs)::[]
234                                                         | Tchar -> ("stderr,
                                                             \"%c\\n\", " ^
                                                            string_of_c_expr
                                                             lhs ^ "*" ^
                                                            string_of_c_expr
                                                             rhs)::[]
235                                                         | Tstring -> ("
                                                            stderr, \"%s\\n
                                                            \", " ^
                                                            string_of_c_expr
                                                             lhs ^ "*" ^
                                                            string_of_c_expr
                                                             rhs)::[]
236                                                         | Tvoid -> raise (
                                                            Error("Invalid
                                                            print type Void:
                                                             " ^
                                                            string_of_c_expr
                                                             lhs ^ "*" ^
                                                            string_of_c_expr
                                                             rhs))
237                                                         | Tfile -> raise (
                                                            Error("Invalid
                                                            print type File:
                                                             ")))
238                                         | _ -> raise (Error("Invalid add in
                                                    function call"))
239                                    )
240                                | Divide -> (match lhs with
241                                             C_Int(l) -> ("stderr, \"%d\\n\", " ^
                                                 string_of_c_expr lhs ^ "/" ^
                                                 string_of_c_expr rhs)::[]
242                                         | C_Float(l) -> ("stderr, \"%f\\n\", " ^
                                                string_of_c_expr lhs ^ "/" ^
                                               string_of_c_expr rhs)::[]
243                                         | C_Id(l, t) -> (match t with
244                                                           Tint | Tintap |
                                                              Tintam -> ("
                                                              stderr, \"%d\\
```

```
                                                                n\", " ^
                                                                 string_of_c_expr
                                                                  lhs ^ "/" ^
                                                                 string_of_c_expr
                                                                  rhs)::[]
245                                                           | Tfloat | Tfloatap
                                                                | Tfloatam -> ("
                                                                stderr, \"%f\\n
                                                                \", " ^
                                                                string_of_c_expr
                                                                 lhs ^ "/" ^
                                                                string_of_c_expr
                                                                 rhs)::[]
246                                                           | Tchar -> ("stderr,
                                                                \"%c\\n\", " ^
                                                                string_of_c_expr
                                                                 lhs ^ "/" ^
                                                                string_of_c_expr
                                                                 rhs)::[]
247                                                           | Tstring -> ("
                                                                stderr, \"%s\\n
                                                                \", " ^
                                                                string_of_c_expr
                                                                 lhs ^ "/" ^
                                                                string_of_c_expr
                                                                 rhs)::[]
248                                                           | Tvoid -> raise (
                                                                Error("Invalid
                                                                print type Void:
                                                                " ^
                                                                string_of_c_expr
                                                                 lhs ^ "/" ^
                                                                string_of_c_expr
                                                                 rhs))
249                                                           | Tfile -> raise (
                                                                Error("Invalid
                                                                print type File:
                                                                ")))
250                                                 | _ -> raise (Error("Invalid add in
                                                        function call"))
251                                               )
252                                         | Equal -> ("stderr, \"%d\\n\", " ^
                                            string_of_c_expr lhs ^ "==" ^ string_of_c_expr
                                             rhs)::[]
253                                         | Ne -> ("stderr, \"%d\\n\", " ^ string_of_c_expr
                                            lhs ^ "!=" ^ string_of_c_expr rhs)::[]
254                                         | Lt -> ("stderr, \"%d\\n\", " ^ string_of_c_expr
                                            lhs ^ "<" ^ string_of_c_expr rhs)::[]
255                                         | Le -> ("stderr, \"%d\\n\", " ^ string_of_c_expr
                                            lhs ^ "<=" ^ string_of_c_expr rhs)::[]
256                                         | Gt -> ("stderr, \"%d\\n\", " ^ string_of_c_expr
                                            lhs ^ ">" ^ string_of_c_expr rhs)::[]
257                                         | Ge -> ("stderr, \"%d\\n\", " ^ string_of_c_expr
                                            lhs ^ ">=" ^ string_of_c_expr rhs)::[]
258                                         | Or -> ("stderr, \"%d\\n\", " ^ string_of_c_expr
                                            lhs ^ "||" ^ string_of_c_expr rhs)::[]
259                                         | And -> ("stderr, \"%d\\n\", " ^ string_of_c_expr
                                             lhs ^ "&&" ^ string_of_c_expr rhs)::[]
```

```
260                                   | Mod -> ("stderr, \"%d\\n\", " ^ string_of_c_expr
                                          lhs ^ "%" ^ string_of_c_expr rhs)::[]
261                                     )
262       | _ -> raise (Error("Invalid expr in print statement: " ^ string_of_c_expr e))
263
264 (*
265 String of stitch expression is for use exclusively inside stitch loops.
266 It takes in a structname and a symbol table in addition to the corresponding
      expression.
267 The structname is used to prepend onto variables that exist inside the symtable (
      which means
268   that the variables are passed into the function externally)
269 *)
270 let rec string_of_stch_expr (structname: string) (table: symTable) (exp: c_expr) =
      match exp with
271     C_Int(l) -> string_of_int l
272   | C_Float(l) -> string_of_float l
273   | C_Char(l) ->  "\'" ^ String.make 1 l ^ "\'"
274   | C_Id(s, t) -> (* structname ^ "->" ^ s *)
275       if List.exists( fun(_,n,_) -> n = s) table.vars then
276           structname ^ "->" ^ s
277       else
278           s
279   | C_Escape(l) -> "\'" ^ l ^ "\'"
280   | C_String(s) -> "\"" ^ s ^ "\""
281   | C_Binop(e1, o, e2) ->
282       (string_of_stch_expr structname table e1) ^ " " ^
283       (match o with
284         Add -> "+" | Subtract -> "-" | Times -> "*" | Divide -> "/"
285       | Equal -> "==" | Ne -> "!="
286       | Lt -> "<" | Le -> "<=" | Gt -> ">" | Ge -> ">="
287       | Or -> "||" | And -> "&&" | Mod -> "%" ) ^ " " ^
288       (string_of_stch_expr structname table e2)
289   | C_Negate(e) -> "!" ^ string_of_stch_expr structname table e
290   | C_Call(f, el) -> (match f with
291         "print" -> "printf"
292         | "error" -> "fprintf"
293         | _ -> f) ^ "(" ^ String.concat ", " (match f with
294           "print" -> print_2_fprint (List.hd el) structname table
295           | "error" -> error_2_fprintf (List.hd el) | _ -> List.map string_of_c_expr
                el) ^ ")"
296   (* Now we need to check to see if the id's are in the table  *)
297   | C_Assign2(i, e) ->
298       if List.exists( fun(_,s,_) -> s = i) table.vars then
299         structname ^ "->" ^ i ^ " = " ^ string_of_stch_expr structname table e
300       else
301         i ^ " = " ^ string_of_stch_expr structname table e
302   | C_Array_Item_Assign(id, ind, e) ->
303       if List.exists( fun(_,s,_) -> s = id) table.vars then
304         structname ^ "->" ^ id ^ "[" ^ string_of_stch_expr structname table ind ^
305         "] = " ^ string_of_stch_expr structname table e
306       else
307         id ^ "[" ^ string_of_stch_expr structname table ind ^
308         "] = " ^ string_of_stch_expr structname table e
309   | C_Array_Index(a, i, t) ->
310       if List.exists( fun(_,s,_) -> s = a) table.vars then
311         structname ^ "->" ^ a ^ "[" ^ string_of_stch_expr structname table i ^ "]"
312       else
```

```
313         a ^ "[" ^ string_of_stch_expr structname table i ^ "]"
314   | C_Matrix_Index(m, r, c, t) ->
315       if List.exists( fun(_,s,_) -> s = m) table.vars then
316         structname ^ "->" ^ m ^ "[" ^ string_of_stch_expr structname table r ^
317         "][" ^ string_of_stch_expr structname table c ^ "]"
318       else
319         m ^ "[" ^ string_of_stch_expr structname table r ^
320         "][" ^ string_of_stch_expr structname table c ^ "]"
321   | C_Matrix_Item_Assign(m, r, c, e) ->
322       if List.exists( fun(_,s,_) -> s = m) table.vars then
323         structname ^ "->" ^ m ^ "[" ^ string_of_stch_expr structname table r ^
324         "][" ^ string_of_stch_expr structname table c ^ "] = " ^ string_of_stch_expr
              structname table e
325       else
326         m ^ "[" ^ string_of_stch_expr structname table r ^
327         "][" ^ string_of_stch_expr structname table c ^ "] = " ^ string_of_stch_expr
              structname table e
328   | C_Noexpr -> ""
329
330       and print_2_fprint (e: c_expr) (structname: string) (table: symTable) = match
              e with
331         C_Int(l) -> ("\"%d\\n\", " ^ (string_of_stch_expr structname table e))::[]
332       | C_Float(l) -> ("\"%f\\n\", " ^ string_of_c_expr e)::[]
333       | C_Char(l) -> ("\"%c\\n\", " ^ string_of_c_expr e)::[]
334       | C_String(l) -> ("\"%s\\n\", " ^ string_of_c_expr e)::[]
335       | C_Array_Index(a, i, t) -> (match t with
336                                      Tint | Tintap | Tintam -> ("\"%d\\n\", " ^ a ^
                                           "[" ^ string_of_c_expr i ^ "]")::[]
337                                    | Tfloat | Tfloatap | Tfloatam -> ("\"%f\\n\", "
                                         ^ a ^ "[" ^ string_of_c_expr i ^ "]")::[]
338                                    | Tchar -> ("\"%c\\n\", " ^ a ^ "[" ^
                                         string_of_c_expr i ^ "]")::[]
339                                    | Tstring -> ("\"%s\\n\", " ^ a ^ "[" ^
                                         string_of_c_expr i ^ "]")::[]
340                                    | Tvoid -> raise (Error("Invalid print type Void
                                         : " ^ a ^ "[" ^ string_of_c_expr i ^ "]"))
341                                    | Tfile -> raise (Error("Invalid print type File
                                         : ")))
342       | C_Matrix_Index(m, r, c, t) -> (match t with
343                                       Tint | Tintap | Tintam -> ("\"%d\\n\", " ^ m ^
                                            "[" ^ string_of_c_expr r ^ "][" ^
344                                                 string_of_c_expr c ^ "]")::[]
345                                    | Tfloat | Tfloatap | Tfloatam -> ("\"%f\\n\", "
                                         ^ m ^ "[" ^ string_of_c_expr r ^ "][" ^
346                                                 string_of_c_expr c ^ "]")::[]
347                                    | Tchar -> ("\"%c\\n\", " ^ m ^ "[" ^
                                         string_of_c_expr r ^ "][" ^
348                                                 string_of_c_expr c ^ "]")::[]
349                                    | Tstring -> ("\"%s\\n\", " ^ m ^ "[" ^
                                         string_of_c_expr r ^ "][" ^
350                                                 string_of_c_expr c ^ "]")::[]
351                                    | Tvoid -> raise(Error("Invalid print type void
                                         in matrix printing"))
352                                    | Tfile -> raise (Error("Invalid print type File
                                         : ")))
353       | C_Id(l, t) -> (match t with
354                           Tint | Tintap | Tintam -> ("\"%d\\n\", " ^ (
                                string_of_stch_expr structname table e))::[]
```

```
355                         | Tfloat | Tfloatap | Tfloatam -> ("\"%f\\n\", " ^
                              string_of_stch_expr structname table e)::[]
356                         | Tchar -> ("\"%c\\n\", " ^ string_of_c_expr e)::[]
357                         | Tstring -> ("\"%s\\n\", " ^ string_of_c_expr e)::[]
358                         | Tvoid -> raise (Error("Invalid print type Void: " ^
                              string_of_c_expr e))
359                         | Tfile -> raise (Error("Invalid print type File: ")))
360       | C_Binop(lhs, o, rhs) -> (match o with
361                                    Add -> (match lhs with
362                                              C_Int(l) -> ("\"%d\\n\", " ^
                                                  string_of_c_expr lhs ^ "+" ^
                                                  string_of_c_expr rhs)::[]
363                                            | C_Float(l) -> ("\"%f\\n\", " ^
                                                string_of_c_expr lhs ^ "+" ^
                                                string_of_c_expr rhs)::[]
364                                            | C_Id(l, t) -> (match t with
365                                                               Tint | Tintap |
                                                                   Tintam ->
                                                                   ("\"%d\\n\", "
                                                                   ^
                                                                   string_of_c_expr
                                                                    lhs ^ "+" ^
                                                                   string_of_c_expr
                                                                    rhs)::[]
366                                                             | Tfloat | Tfloatap
                                                                | Tfloatam ->
                                                                ("\"%f\\n\", " ^

                                                                string_of_c_expr
                                                                 lhs ^ "+" ^
                                                                string_of_c_expr
                                                                 rhs)::[]
367                                                             | Tchar -> ("\"%c\\n
                                                                \", " ^
                                                                string_of_c_expr
                                                                 lhs ^ "+" ^
                                                                string_of_c_expr
                                                                 rhs)::[]
368                                                             | Tstring -> ("\"%s
                                                                \\n\", " ^
                                                                string_of_c_expr
                                                                 lhs ^ "+" ^
                                                                string_of_c_expr
                                                                 rhs)::[]
369                                                             | Tvoid -> raise (
                                                                Error("Invalid
                                                                print type Void:
                                                                 " ^
                                                                string_of_c_expr
                                                                 lhs ^ "+" ^
                                                                string_of_c_expr
                                                                 rhs))
370                                                             | Tfile -> raise (
                                                                Error("Invalid
                                                                print type File:
                                                                 ")))
371                                            | _ -> raise (Error("Invalid add in
                                                function call")))
```

```
372                                     )
373                                 | Subtract -> (match lhs with
374                                             C_Int(l) -> ("\"%d\\n\", " ^
                                                string_of_c_expr lhs ^ "-" ^
                                                string_of_c_expr rhs)::[]
375                                           | C_Float(l) -> ("\"%f\\n\", " ^
                                              string_of_c_expr lhs ^ "-" ^
                                              string_of_c_expr rhs)::[]
376                                           | C_Id(l, t) -> (match t with
377                                                                 Tint | Tintap |
                                                                    Tintam ->
                                                                    ("\"%d\\n\", "
                                                                     ^
                                                                    string_of_c_expr
                                                                     lhs ^ "-" ^
                                                                    string_of_c_expr
                                                                     rhs)::[]
378                                                               | Tfloat | Tfloatap
                                                                  | Tfloatam ->
                                                                  ("\"%f\\n\", " ^

                                                                  string_of_c_expr
                                                                   lhs ^ "-" ^
                                                                  string_of_c_expr
                                                                   rhs)::[]
379                                                               | Tchar -> ("\"%c\\n
                                                                  \", " ^
                                                                  string_of_c_expr
                                                                   lhs ^ "-" ^
                                                                  string_of_c_expr
                                                                   rhs)::[]
380                                                               | Tstring -> ("\"%s
                                                                  \\n\", " ^
                                                                  string_of_c_expr
                                                                   lhs ^ "-" ^
                                                                  string_of_c_expr
                                                                   rhs)::[]
381                                                               | Tvoid -> raise (
                                                                  Error("Invalid
                                                                  print type Void:
                                                                   " ^
                                                                  string_of_c_expr
                                                                   lhs ^ "-" ^
                                                                  string_of_c_expr
                                                                   rhs))
382                                                               | Tfile -> raise (
                                                                  Error("Invalid
                                                                  print type File:
                                                                   ")))
383                                           | _ -> raise (Error("Invalid add in
                                              function call"))
384                                     )
385                                 | Times -> (match lhs with
386                                             C_Int(l) -> ("\"%d\\n\", " ^
                                                string_of_c_expr lhs ^ "*" ^
                                                string_of_c_expr rhs)::[]
387                                           | C_Float(l) -> ("\"%f\\n\", " ^
                                              string_of_c_expr lhs ^ "*" ^
```

```
                                        string_of_c_expr rhs)::[]
388                                   | C_Id(l, t) -> (match t with
389                                                       Tint | Tintap |
                                                            Tintam ->
                                                            ("\"%d\\n\", "
                                                             ^
                                                            string_of_c_expr
                                                             lhs ^ "*" ^
                                                            string_of_c_expr
                                                             rhs)::[]
390                                                     | Tfloat | Tfloatap
                                                          | Tfloatam ->
                                                          ("\"%f\\n\", " ^

                                                          string_of_c_expr
                                                           lhs ^ "*" ^
                                                          string_of_c_expr
                                                           rhs)::[]
391                                                     | Tchar -> ("\"%c\\n
                                                          \", " ^
                                                          string_of_c_expr
                                                           lhs ^ "*" ^
                                                          string_of_c_expr
                                                           rhs)::[]
392                                                     | Tstring -> ("\"%s
                                                          \\n\", " ^
                                                          string_of_c_expr
                                                           lhs ^ "*" ^
                                                          string_of_c_expr
                                                           rhs)::[]
393                                                     | Tvoid -> raise (
                                                          Error("Invalid
                                                          print type Void:
                                                           " ^
                                                          string_of_c_expr
                                                           lhs ^ "*" ^
                                                          string_of_c_expr
                                                           rhs))
394                                                     | Tfile -> raise (
                                                          Error("Invalid
                                                          print type File:
                                                           ")))
395                                   | _ -> raise (Error("Invalid add in
                                          function call"))
396                                )
397                             | Divide -> (match lhs with
398                                          C_Int(l) -> ("\"%d\\n\", " ^
                                              string_of_c_expr lhs ^ "/" ^
                                              string_of_c_expr rhs)::[]
399                                        | C_Float(l) -> ("\"%f\\n\", " ^
                                            string_of_c_expr lhs ^ "/" ^
                                            string_of_c_expr rhs)::[]
400                                        | C_Id(l, t) -> (match t with
401                                                            Tint | Tintap |
                                                                 Tintam ->
                                                                 ("\"%d\\n\", "
                                                                  ^
                                                                 string_of_c_expr
```

```
                                                                    lhs ^ "/" ^
                                                                   string_of_c_expr
                                                                    rhs)::[]
402                                                              | Tfloat | Tfloatap
                                                                   | Tfloatam ->
                                                                   ("\"%f\\n\", " ^

                                                                   string_of_c_expr
                                                                    lhs ^ "/" ^
                                                                   string_of_c_expr
                                                                    rhs)::[]
403                                                              | Tchar -> ("\"%c\\n
                                                                   \", " ^
                                                                   string_of_c_expr
                                                                    lhs ^ "/" ^
                                                                   string_of_c_expr
                                                                    rhs)::[]
404                                                              | Tstring -> ("\"%s
                                                                   \\n\", " ^
                                                                   string_of_c_expr
                                                                    lhs ^ "/" ^
                                                                   string_of_c_expr
                                                                    rhs)::[]
405                                                              | Tvoid -> raise (
                                                                   Error("Invalid
                                                                   print type Void:
                                                                    " ^
                                                                   string_of_c_expr
                                                                    lhs ^ "/" ^
                                                                   string_of_c_expr
                                                                    rhs))
406                                                              | Tfile -> raise (
                                                                   Error("Invalid
                                                                   print type File:
                                                                    ")))
407                                                 | _ -> raise (Error("Invalid add in
                                                        function call"))
408                                          )
409                                    | Equal -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                         "==" ^ string_of_c_expr rhs)::[]
410                                    | Ne -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                         "!=" ^ string_of_c_expr rhs)::[]
411                                    | Lt -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                         "<" ^ string_of_c_expr rhs)::[]
412                                    | Le -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                         "<=" ^ string_of_c_expr rhs)::[]
413                                    | Gt -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                         ">" ^ string_of_c_expr rhs)::[]
414                                    | Ge -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                         ">=" ^ string_of_c_expr rhs)::[]
415                                    | Or -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                         "||" ^ string_of_c_expr rhs)::[]
416                                    | And -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                         "&&" ^ string_of_c_expr rhs)::[]
417                                    | Mod -> ("\"%d\\n\", " ^ string_of_c_expr lhs ^
                                         "%" ^ string_of_c_expr rhs)::[]
418                                          )
419      | _ -> raise (Error("Invalid expr in print statement: " ^ string_of_c_expr e))
```

```
420
421     and error_2_fprintf (e: c_expr) = match e with
422       C_Int(l) -> ("stderr, \"%d\\n\", " ^ string_of_c_expr e)::[]
423     | C_Float(l) -> ("stderr, \"%f\\n\", " ^ string_of_c_expr e)::[]
424     | C_Char(l) -> ("stderr, \"%c\\n\", " ^ string_of_c_expr e)::[]
425     | C_String(l) -> ("stderr, \"%s\\n\", " ^ string_of_c_expr e)::[]
426     | C_Id(l, t) -> (match t with
427                         Tint | Tintap | Tintam -> ("stderr, \"%d\\n\", " ^
                              string_of_c_expr e)::[]
428                       | Tfloat | Tfloatap | Tfloatam -> ("stderr, \"%f\\n\", " ^
                          string_of_c_expr e)::[]
429                       | Tchar -> ("stderr, \"%c\\n\", " ^ string_of_c_expr e)::[]
430                       | Tstring -> ("stderr, \"%s\\n\", " ^ string_of_c_expr e)
                          ::[]
431                       | Tvoid -> raise (Error("Invalid print type Void: " ^
                          string_of_c_expr e))
432                       | Tfile -> raise (Error("Invalid print type File: ")))
433     | C_Binop(lhs, o, rhs) -> (match o with
434                                   Add -> (match lhs with
435                                             C_Int(l) -> ("stderr, \"%d\\n\", " ^
                                                 string_of_c_expr lhs ^ "+" ^
                                                 string_of_c_expr rhs)::[]
436                                           | C_Float(l) -> ("stderr, \"%f\\n\", " ^
                                               string_of_c_expr lhs ^ "+" ^
                                              string_of_c_expr rhs)::[]
437                                           | C_Id(l, t) -> (match t with
438                                                              Tint | Tintap |
                                                                  Tintam -> ("
                                                                  stderr, \"%d\\
                                                                  n\", " ^
                                                                  string_of_c_expr
                                                                   lhs ^ "+" ^
                                                                  string_of_c_expr
                                                                   rhs)::[]
439                                                            | Tfloat | Tfloatap
                                                                | Tfloatam -> ("
                                                               stderr, \"%f\\n
                                                               \", " ^
                                                               string_of_c_expr
                                                                lhs ^ "+" ^
                                                               string_of_c_expr
                                                                rhs)::[]
440                                                            | Tchar -> ("stderr,
                                                                \"%c\\n\", " ^
                                                               string_of_c_expr
                                                                lhs ^ "+" ^
                                                               string_of_c_expr
                                                                rhs)::[]
441                                                            | Tstring -> ("
                                                               stderr, \"%s\\n
                                                               \", " ^
                                                               string_of_c_expr
                                                                lhs ^ "+" ^
                                                               string_of_c_expr
                                                                rhs)::[]
442                                                            | Tvoid -> raise (
                                                               Error("Invalid
                                                               print type Void:
```

```
                                                                    " ^
                                                                   string_of_c_expr
                                                                    lhs ^ "+" ^
                                                                   string_of_c_expr
                                                                    rhs))
443                                                              | Tfile -> raise (
                                                                   Error("Invalid
                                                                   print type File:
                                                                    ")))
444                                                  | _ -> raise (Error("Invalid add in
                                                         function call"))
445                                              )
446                                          | Subtract -> (match lhs with
447                                                C_Int(l) -> ("stderr, \"%d\\n\", " ^
                                                    string_of_c_expr lhs ^ "-" ^
                                                    string_of_c_expr rhs)::[]
448                                                | C_Float(l) -> ("stderr, \"%f\\n\", " ^
                                                    string_of_c_expr lhs ^ "-" ^
                                                    string_of_c_expr rhs)::[]
449                                                | C_Id(l, t) -> (match t with
450                                                                   Tint | Tintap |
                                                                       Tintam -> ("
                                                                       stderr, \"%d\\
                                                                       n\", " ^
                                                                       string_of_c_expr
                                                                        lhs ^ "-" ^
                                                                       string_of_c_expr
                                                                        rhs)::[]
451                                                              | Tfloat | Tfloatap
                                                                   | Tfloatam -> ("
                                                                   stderr, \"%f\\n
                                                                   \", " ^
                                                                   string_of_c_expr
                                                                    lhs ^ "-" ^
                                                                   string_of_c_expr
                                                                    rhs)::[]
452                                                              | Tchar -> ("stderr,
                                                                    \"%c\\n\", " ^
                                                                   string_of_c_expr
                                                                    lhs ^ "-" ^
                                                                   string_of_c_expr
                                                                    rhs)::[]
453                                                              | Tstring -> ("
                                                                   stderr, \"%s\\n
                                                                   \", " ^
                                                                   string_of_c_expr
                                                                    lhs ^ "-" ^
                                                                   string_of_c_expr
                                                                    rhs)::[]
454                                                              | Tvoid -> raise (
                                                                   Error("Invalid
                                                                   print type Void:
                                                                    " ^
                                                                   string_of_c_expr
                                                                    lhs ^ "-" ^
                                                                   string_of_c_expr
                                                                    rhs))
455                                                              | Tfile -> raise (
```

```
                                                    Error("Invalid
                                                    print type File:
                                                    ")))
456                                 | _ -> raise (Error("Invalid add in
                                        function call"))
457                             )
458                         | Times -> (match lhs with
459                                 C_Int(l) -> ("stderr, \"%d\\n\", " ^
                                        string_of_c_expr lhs ^ "*" ^
                                        string_of_c_expr rhs)::[]
460                               | C_Float(l) -> ("stderr, stderr, \"%f\\
                                    n\", " ^ string_of_c_expr lhs ^ "*"
                                    ^ string_of_c_expr rhs)::[]
461                               | C_Id(l, t) -> (match t with
462                                                   Tint | Tintap |
                                                       Tintam -> ("
                                                       stderr, \"%d\\
                                                       n\", " ^
                                                       string_of_c_expr
                                                        lhs ^ "*" ^
                                                       string_of_c_expr
                                                        rhs)::[]
463                                                 | Tfloat | Tfloatap
                                                     | Tfloatam -> ("
                                                     stderr, \"%f\\n
                                                     \", " ^
                                                     string_of_c_expr
                                                      lhs ^ "*" ^
                                                     string_of_c_expr
                                                      rhs)::[]
464                                                 | Tchar -> ("stderr,
                                                     \"%c\\n\", " ^
                                                     string_of_c_expr
                                                      lhs ^ "*" ^
                                                     string_of_c_expr
                                                      rhs)::[]
465                                                 | Tstring -> ("
                                                     stderr, \"%s\\n
                                                     \", " ^
                                                     string_of_c_expr
                                                      lhs ^ "*" ^
                                                     string_of_c_expr
                                                      rhs)::[]
466                                                 | Tvoid -> raise (
                                                     Error("Invalid
                                                     print type Void:
                                                     " ^
                                                     string_of_c_expr
                                                      lhs ^ "*" ^
                                                     string_of_c_expr
                                                      rhs))
467                                                 | Tfile -> raise (
                                                     Error("Invalid
                                                     print type File:
                                                     ")))
468                                 | _ -> raise (Error("Invalid add in
                                        function call"))
469                             )
```

```
470                                 | Divide -> (match lhs with
471                                             C_Int(l) -> ("stderr, \"%d\\n\", " ^
                                                string_of_c_expr lhs ^ "/" ^
                                                string_of_c_expr rhs)::[]
472                                           | C_Float(l) -> ("stderr, \"%f\\n\", " ^
                                              string_of_c_expr lhs ^ "/" ^
                                             string_of_c_expr rhs)::[]
473                                           | C_Id(l, t) -> (match t with
474                                                                Tint | Tintap |
                                                                    Tintam -> ("
                                                                    stderr, \"%d\\
                                                                    n\", " ^
                                                                    string_of_c_expr
                                                                     lhs ^ "/" ^
                                                                    string_of_c_expr
                                                                     rhs)::[]
475                                                              | Tfloat | Tfloatap
                                                                  | Tfloatam -> ("
                                                                  stderr, \"%f\\n
                                                                  \", " ^
                                                                  string_of_c_expr
                                                                   lhs ^ "/" ^
                                                                  string_of_c_expr
                                                                   rhs)::[]
476                                                              | Tchar -> ("stderr,
                                                                   \"%c\\n\", " ^
                                                                  string_of_c_expr
                                                                   lhs ^ "/" ^
                                                                  string_of_c_expr
                                                                   rhs)::[]
477                                                              | Tstring -> ("
                                                                  stderr, \"%s\\n
                                                                  \", " ^
                                                                  string_of_c_expr
                                                                   lhs ^ "/" ^
                                                                  string_of_c_expr
                                                                   rhs)::[]
478                                                              | Tvoid -> raise (
                                                                  Error("Invalid
                                                                  print type Void:
                                                                   " ^
                                                                  string_of_c_expr
                                                                   lhs ^ "/" ^
                                                                  string_of_c_expr
                                                                   rhs))
479                                                              | Tfile -> raise (
                                                                  Error("Invalid
                                                                  print type File:
                                                                   ")))
480                                           | _ -> raise (Error("Invalid add in
                                                 function call"))
481                                      )
482                                 | Equal -> ("stderr, \"%d\\n\", " ^
                                    string_of_c_expr lhs ^ "==" ^ string_of_c_expr
                                     rhs)::[]
483                                 | Ne -> ("stderr, \"%d\\n\", " ^ string_of_c_expr
                                    lhs ^ "!=" ^ string_of_c_expr rhs)::[]
484                                 | Lt -> ("stderr, \"%d\\n\", " ^ string_of_c_expr
```

```
                                        lhs ^ "<" ^ string_of_c_expr rhs)::[]
485                                   | Le -> ("stderr, \"%d\\n\", " ^ string_of_c_expr
                                        lhs ^ "<=" ^ string_of_c_expr rhs)::[]
486                                   | Gt -> ("stderr, \"%d\\n\", " ^ string_of_c_expr
                                        lhs ^ ">" ^ string_of_c_expr rhs)::[]
487                                   | Ge -> ("stderr, \"%d\\n\", " ^ string_of_c_expr
                                        lhs ^ ">=" ^ string_of_c_expr rhs)::[]
488                                   | Or -> ("stderr, \"%d\\n\", " ^ string_of_c_expr
                                        lhs ^ "||" ^ string_of_c_expr rhs)::[]
489                                   | And -> ("stderr, \"%d\\n\", " ^ string_of_c_expr
                                         lhs ^ "&&" ^ string_of_c_expr rhs)::[]
490                                   | Mod -> ("stderr, \"%d\\n\", " ^ string_of_c_expr
                                         lhs ^ "%" ^ string_of_c_expr rhs)::[]
491                                    )
492       | _ -> raise (Error("Invalid expr in print statement: " ^ string_of_c_expr e))
493
494
495
496 let string_of_c_vdecl vdecl = string_of_c_dataType vdecl.vdecl_type ^ " " ^ vdecl.
      vdecl_name (* " " ^ vdecl.array_size ^ *)
497
498 let string_of_c_arraydecl arraydecl = string_of_c_dataType arraydecl.arraydecl_type
      ^ " " ^ arraydecl.arraydecl_name ^ "[" ^
499     string_of_expr arraydecl.arraydecl_size ^ "]"
500
501 (* print stitch variables will generate the C code to put the necessary variables
502    into the struct that gets passed to the pthread function. Most of these are
         straightforward,
503    just copying the name into the struct. For matrices and arrays, we need to
         generate special
504    pointer notation since we aren't copying these like the other variables
505     *)
506 let rec print_stitch_variables (seed: string) el = match el with
507   [] -> seed ^ "\n"
508   | head::tail -> let (typ, name, exp) = head in
509     if exp = C_Noexpr then
510       print_stitch_variables (seed ^ (string_of_dataType typ) ^ " " ^ name ^
            string_of_c_expr exp ^ ";\n") tail
511     else (match exp with
512       | C_Matrix_Index(nm,ro,col,dt) -> print_stitch_variables (seed ^ (
            string_of_dataType typ) ^
513         " (* " ^ name ^ ")["^string_of_c_expr col^"];\n") tail
514       | C_Array_Index(name, exp,typ) -> print_stitch_variables (seed ^
515         (string_of_dataType typ) ^ " *" ^ name ^ ";\n") tail
516       | _ -> raise(Error("How did we even get here?")) )
517
518 (* Assign stitch variables is like print_stitch_variables, except it generates the C
       code to assign
519   the local variables into their counterparts in the structure that's passed in
520   It uses the same list and generates the same variables
521    *)
522 let rec assign_stitch_variables (seed: string) (structname: string) el = match el
      with
523   [] -> seed ^ "\n"
524   | head::tail -> let (typ, name, exp) = head in
525     assign_stitch_variables (seed ^ structname ^ "." ^ name ^ " = " ^ name ^ ";\n")
         (structname) tail
526
```

```
527 (* This generates the loop after each stitch loops that will resolve the accumulator
        variables.
528 Right now this only works with int accumulators, as accumulators are unfinished at
       the time
529 of this submission
530  *)
531 let rec resolve_accums (seed: string) (structname: string ) el = match el with
532   [] -> seed ^ "\n"
533   | head::tail -> let (typ, name, exp) = head in
534     (match typ with
535       (Tintap | Tfloatap) -> resolve_accums (seed ^ name ^ "+=" ^ structname ^ "." ^
              name ^";\n") structname tail
536       |_ -> resolve_accums seed structname tail)
537
538
539 let rec string_of_c_matrixlist (seed: string) el = match el with
540     [] -> seed ^ "}"
541     | head::tail -> string_of_c_matrixlist (seed ^ string_of_arraylist head ^ ",\n")
            tail
542
543 let string_of_c_matrixdecl m = string_of_c_dataType m.matrixdecl_type ^ " " ^ m.
       matrixdecl_name ^ "[" ^
544     string_of_expr m.matrixdecl_rows ^ "][" ^ string_of_expr m.matrixdecl_cols ^ "]"
545
546 (* Converts a stitch loop into a for loop that creates all the threading information
       .
547   Allocates the threadpool and the structpool, using the procedurally generated
        function suffix
548    *)
549 let convert_stitch_2_for var start s_end stride fname scope =
550   let size = string_of_c_expr s_end in
551   let threads = "\npthread_t *threadpool" ^ fname ^ " = malloc(NUMTHREADS * sizeof(
        pthread_t));\n" in
552
553   (* Assign the initial variables into the struct *)
554   let thread_assignment = "info"^fname^"[thread"^fname^"].begin = i;\n" ^
555                                 (assign_stitch_variables "" ("info"^fname^"[thread
                                      "^fname^"]") scope.vars )^
556                                 "if((" ^ string_of_c_expr var ^ " + 2*(" ^ size ^
                                      "/NUMTHREADS)) > " ^ size ^ ") {\n" ^
557                                 "info"^fname^"[thread"^fname^"].end = " ^ size ^
                                      ";\n" ^
558                                 string_of_c_expr var ^ " = " ^ size ^ ";\n" ^
559                                 "}\n" ^
560                                 "else {\n" ^
561                                 "info"^fname^"[thread"^fname^"].end = " ^
                                      string_of_c_expr var ^ " + " ^ size ^ "/
                                      NUMTHREADS;\n" ^
562                                 "}\n" in
563
564   (* Code to generate the threadpool *)
565   let threadgen = "int e = pthread_create(&threadpool"^fname^"[thread"^fname^"],
        NULL, " ^ fname ^ ", &info"^fname^"[thread"^fname^"]);\n" ^
566                   "if (e != 0) {\n" ^
567                   "perror(\"Cannot create thread!\");\n" ^
568                   "free(threadpool"^fname^"); //error, free the threadpool\n" ^
569                   "exit(1);\n" ^
570                   "}\n" in
```

```
571
572   (* Code that blocks and waits for the threads to finish *)
573   let threadjoin = "//loop and wait for all the threads to finish\n" ^
574                    "for(" ^ string_of_c_expr var ^ " = 0; "^ string_of_c_expr var ^
575                    " < NUMTHREADS; "^string_of_c_expr var ^"++) {\n" ^
576                    "pthread_join(threadpool"^fname^"["^ string_of_c_expr var ^"],
                         NULL);\n" ^
577                    "}\n" in
578
579   (* The loop at the end to resolve any accumulators, if they were used *)
580   let accums = "//now we loop and resolve any accumulators\n" ^
581                    "for(" ^ string_of_c_expr var ^ " = 0; "^ string_of_c_expr var ^
582                    " < NUMTHREADS; "^string_of_c_expr var ^"++) {\n" ^
583                    (resolve_accums "" ("info"^fname^"["^string_of_c_expr var^"]")
                         scope.vars) ^
584                    "}\n\n" in
585
586   let varinfo = "struct stch_rangeInfo" ^ fname ^ " *info"^fname^" = malloc(sizeof(
       struct stch_rangeInfo" ^fname^") * NUMTHREADS);\n" in
587   let incr = string_of_c_expr s_end ^ "/" ^ "NUMTHREADS" in
588   let loop = threads ^ varinfo ^ "int thread"^fname^" = 0;\n" ^ "for(" in
589   loop ^ string_of_c_expr var ^ " = " ^ string_of_c_expr start ^ ";" ^
         string_of_c_expr var ^ " < " ^
590     string_of_c_expr s_end ^ ";" ^ string_of_c_expr var ^ " = " ^ string_of_c_expr
          var ^ "+" ^ incr ^
591     ") {\n" ^ thread_assignment ^ threadgen ^ "thread"^fname^"++;\n" ^ "}\n\n" ^
          threadjoin ^ accums
592
593 (* String of c statements. The optional variable here is not ever used, but I'm
       afraid to take it out
594   right before we submit in case it breaks anything
595    *)
596 let rec string_of_c_stmt ?structname:(structname="") (st: c_stmt)= match st with
597     C_Block(_, stmts) ->
598       "{\n" ^ String.concat "" (List.map (string_of_c_stmt ~structname:"hello")
              stmts) ^ "}\n"
599   | C_Expr(_, e) -> string_of_c_expr e ^ ";\n";
600   | C_Vdecl(v) -> string_of_c_dataType v.vdecl_type ^ " " ^ v.vdecl_name ^ ";\n";
601   | C_Return(_, c_expr) -> "return " ^ string_of_c_expr c_expr ^ ";\n";
602   | C_If(e, s, C_Block(_, [])) -> "if (" ^ string_of_c_expr e ^ ")\n" ^
        string_of_c_stmt s
603   | C_If(e, s1, s2) ->  "if (" ^ string_of_c_expr e ^ ")\n" ^
604       string_of_c_stmt s1 ^ "else\n" ^ string_of_c_stmt s2
605   | C_For(e1, e2, e3, s) ->
606       "for (" ^ string_of_c_expr e1  ^ " ; " ^ string_of_c_expr e2 ^ " ; " ^
607       string_of_c_expr e3  ^ ") " ^ string_of_c_stmt s
608   | C_While(e, s) -> "while (" ^ string_of_c_expr e ^ ") " ^ string_of_c_stmt s
609   | C_Stitch(var, start, s_end, stride, fname, body, scope) -> convert_stitch_2_for
        var start s_end stride fname scope
610   | C_Assign(v, e) -> string_of_c_vdecl v ^ " = " ^ string_of_c_expr e ^ ";\n"
611   | C_ArrayDecl(a) -> string_of_c_arraydecl a ^ ";\n"
612   | C_ArrayInit(arraydecl, el) -> string_of_c_arraydecl arraydecl ^ " = {" ^ String.
        concat ", " (List.map string_of_expr el) ^ "};\n"
613   | C_MatrixDecl(m) -> string_of_c_matrixdecl m ^ ";\n"
614   | C_MatrixInit(mdecl, li) -> string_of_c_matrixdecl mdecl ^ " = " ^
        string_of_c_matrixlist "{" li ^ ";\n"
615   | C_Break -> "break;"
616
```

```
617
618 (* This function will take in a structname, a symtable, and a list of statements.
619    It will check to see if the statements need to be prepended with the structname
         by
620    checking the symtable, and do so if it needs to
621    This function is only for stitch loops *)
622
623  let rec string_of_stch_stmt (structname: string) (table: symTable) (st: c_stmt) =
       match st with
624      C_Block(_, stmts) ->
625        "{\n" ^ String.concat "" (List.map (string_of_stch_stmt structname table)
             stmts) ^ "}\n"
626   | C_Expr(_, e) -> string_of_stch_expr structname table e ^ ";\n";
627   | C_Vdecl(v) -> (* string_of_c_dataType v.vdecl_type ^ " " ^ v.vdecl_name ^ ";\n";
         *)
628        if List.exists( fun(_,s,_) -> s = v.vdecl_name) table.vars then
629            string_of_c_dataType v.vdecl_type ^ " " ^ structname ^ "->" ^ v.vdecl_name
                 ^ ";\n"
630        else
631            string_of_c_dataType v.vdecl_type ^ " " ^ v.vdecl_name ^ ";\n"
632   | C_Return(_, c_expr) -> "return " ^ string_of_c_expr c_expr ^ ";\n";
633   | C_If(e, s, C_Block(_, [])) -> "if (" ^ string_of_stch_expr structname table e ^
634       ")\n" ^ string_of_stch_stmt structname table s
635   | C_If(e, s1, s2) ->  "if (" ^ string_of_stch_expr structname table e ^ ")\n" ^
636       string_of_stch_stmt structname table s1 ^ "else\n" ^ string_of_stch_stmt
             structname table s2
637   | C_For(e1, e2, e3, s) ->
638       "for (" ^ string_of_stch_expr structname table e1  ^ " ; " ^
             string_of_stch_expr structname table e2 ^
639         " ; " ^ string_of_stch_expr structname table e3  ^ ") " ^
               string_of_stch_stmt structname table s
640   | C_While(e, s) -> "while (" ^ string_of_stch_expr structname table e ^ ") " ^
       string_of_stch_stmt structname table s
641   | C_Stitch(var, start, s_end, stride, fname, body, scope) -> convert_stitch_2_for
       var start s_end stride fname scope
642
643   (* Assign doesn't need to be checked, it is a variable declaration *)
644   | C_Assign(v, e) ->
645       string_of_c_vdecl v ^ " = " ^ string_of_stch_expr structname table e ^ ";\n"
646
647   (* Array declarations don't need to be checked for struct addition *)
648   | C_ArrayDecl(a) ->
649       string_of_c_arraydecl a ^ ";\n"
650
651   (* Array inits do not need to be checked for symtable locations *)
652   | C_ArrayInit(a, el) ->
653       string_of_c_arraydecl a ^ " = {" ^ String.concat ", " (List.map string_of_expr
             el) ^ "};\n"
654
655   (* Matrix declarations don't need to be checked *)
656   | C_MatrixDecl(m) -> (* string_of_c_matrixdecl m ^ ";\n" *)
657       string_of_c_matrixdecl m ^ ";\n"
658
659   | C_MatrixInit(mdecl, li) -> string_of_c_matrixdecl mdecl ^ " = " ^
       string_of_c_matrixlist "{" li ^ ";\n"
660   | C_Break -> "break;"
661
662
```

```
663 (* Stitch to func will turn the contents of the stitch loop into a function that is
        passed through
664   to each thread. This will properly generate the for loop that runs at the top of
        the function,
665   with each thread starting and ending at locations determined by the initial
        division of labor
666    *)
667 let rec stitch2func = function
668     C_Block(_, stmts) ->
669       String.concat "" (List.map stitch2func stmts)
670   | C_If(e, s, C_Block(_, [])) -> stitch2func s
671   | C_If(e, s1, s2) -> stitch2func s2
672   | C_For(e1, e2, e3, s) -> stitch2func s
673   | C_While(e, s) -> stitch2func s
674   | C_Stitch(var, start, s_end, stride, fname, body, scope) ->
675       let inner = String.concat "\n" (List.map ((string_of_stch_stmt ("((struct
              stch_rangeInfo" ^ fname ^ " *)vars)")) scope) body) in
676       "struct stch_rangeInfo" ^ fname ^ " {\n" ^ "int begin;\n"^ "int end;\n" ^ "int
             stepSize;\n" ^
677       (print_stitch_variables "" scope.vars) ^ "\n};\n\n" ^ "void *" ^ fname ^ " (
             void *vars) {\n " ^
678       "int "^(string_of_c_expr var)^" = 0;\n for("^(string_of_c_expr var)^" = ((
             struct stch_rangeInfo"^
679       fname^" *)vars)->begin; "^(string_of_c_expr var)^" < ((struct stch_rangeInfo"^
             fname^
680         " *)vars)->end; "^(string_of_c_expr var)^"++) {\n" ^ inner ^ "\n}\nreturn (
               void*)0;\n}\n"
681   | _ -> ""
682
683 let string_of_stitch func = String.concat "" (List.map stitch2func func.body)
684
685 let string_of_c_fdecl fdecl = match fdecl.fdecl_name with
686   "main" -> ""
687   | _ -> string_of_c_dataType fdecl.fdecl_type ^ " " ^ fdecl.fdecl_name ^ "(" ^
688     String.concat ", " (List.map string_of_c_vdecl fdecl.fdecl_formals) ^ ")\n{\n" ^
689     String.concat "" (List.map string_of_c_stmt fdecl.body) ^ "}\n"
690
691
692 let string_of_main fdecl = match fdecl.fdecl_name with
693   "main" -> string_of_c_dataType fdecl.fdecl_type ^ " " ^ fdecl.fdecl_name ^ "(" ^
694     String.concat ", " (List.map string_of_c_vdecl fdecl.fdecl_formals) ^ ")\n{\n" ^
695     String.concat "" (List.map string_of_c_stmt fdecl.body) ^ "}\n"
696   | _ -> ""
697
698 let string_of_vars (_, s, _) = s
699
700 let string_of_c_program (prog : Stch_cast.c_program ) =
701   String.concat "" (List.map string_of_c_stmt prog.stmts) ^ "\n" ^
702   String.concat "\n" (List.map string_of_c_fdecl prog.funcs) ^ "\n" ^
703   String.concat "\n" (List.map string_of_stitch prog.funcs) ^ "\n" ^
704   String.concat "\n" (List.map string_of_main prog.funcs) ^ "\n"
```

stitch.ml

```
1 (*-----------------------------------------------------*)
2 (* Parse and print the program                         *)
3 (*-----------------------------------------------------*)
4
5 let filename = Sys.argv.(1) ^ ".c" in
6
7 let in_channel = open_in Sys.argv.(1) in
8
9 let lexbuf = Lexing.from_channel in_channel in
10
11 let program = Stch_parser.program Stch_scanner.token lexbuf in
12 let finalcast =  Stch_semantic.check_prog program in
13 let outprog = C_generator.string_of_c_program finalcast in
14
15 let headers = "#include \"stch_headers.h\"\n\n" ^ outprog
16     in Printf.fprintf (open_out filename) "%s" headers
```

Makefile

```
OBJS = stch_ast.cmo stch_parser.cmo stch_scanner.cmo stch_semantic.cmo c_generator.cmo stitch.cmo

YACC = ocamlyacc

stitch: $(OBJS)
  ocamlc -o stitch $(OBJS)

stch_scanner.ml: stch_scanner.mll
  ocamllex stch_scanner.mll

stch_parser.ml stch_parser.mli: stch_parser.mly
  $(YACC) -v stch_parser.mly

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

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

.PHONY: clean
clean:
  rm -f stitch stch_parser.ml stch_parser.mli stch_scanner.ml \
      *.cmo *.cmi *.out *.diff *.output stitch *.dSYM

.PHONY: all
all: clean stitch

c_generator.cmo : stch_cast.cmi stch_ast.cmo
c_generator.cmx : stch_cast.cmi stch_ast.cmx
stch_ast.cmo :
stch_ast.cmx :
stch_parser.cmo : stch_ast.cmo stch_parser.cmi
stch_parser.cmx : stch_ast.cmx stch_parser.cmi
stch_scanner.cmo : stch_parser.cmi
stch_scanner.cmx : stch_parser.cmx
stch_semantic.cmo : stch_cast.cmi stch_ast.cmo
stch_semantic.cmx : stch_cast.cmi stch_ast.cmx
stitch.cmo :
stitch.cmx :
stch_parser.cmi : stch_ast.cmo
stch_cast.cmi : stch_ast.cmo
```

stch_ptestSuite.sh

```
#!/bin/sh

#Stitch Lang Regression Test Suite for Parser
#
# Author: Megan Skrypek

COL='\033[0;34m'    #Blue color for description
SUCC='\033[1;32m'   #Green color for success
FAIL='\033[0;31m'   #Red color for failure
NC='\033[0m'       #No color - to clear the color after

STITCH="./../ocaml/stitch"
DECTESTS="./_ptests/dec*"
FUNCTESTS="./_ptests/fun*"
LOOPTESTS="./_ptests/loop*"

#print whether we succeeded or failed the test
function echoResult {

  if [ $1 -eq 0 ]; then
    echo "${SUCC}TEST SUCCESSFUL!${NC}"
  else
    echo "${FAIL}TEST FAILED!${NC}"
  fi
}

#print the information about each tests
function printTest {

  echo $COL$(head -n 1 $1) $NC
}


#run all tests based on path passed in
function runTests {

  for test in $@
  do
    echo "Starting test $test"
    printTest $test
    $STITCH $test
    echoResult $?
    echo "\n"
  done


}

#-----------------------------------#
#SCRIPT STARTS HERE                 #
#-----------------------------------#

#Make the compiler if it isn't already made
echo "Making the compiler..."
```

```
55 cd ../ocaml
56 make all > /dev/null
57 cd ../testing
58
59
60 echo "Starting Stitch parse test suite"
61 echo "\n"
62
63 echo "Declaration Tests" #declaration tests
64 runTests $DECTESTS
65 echo "Function Tests" #function tests
66 runTests $FUNCTESTS
67 echo "Loop Tests"      #loop tests
68 runTests $LOOPTESTS
69
70
71 rm _ptests/*.c
```

stch_testSuite.sh

```
#!/bin/sh
#Stitch language regression test suite
#
#Author: Dan Cole
#

COL='\033[0;34m'    #Blue color for description
SUCC='\033[1;32m'   #Green color for success
FAIL='\033[0;31m'   #Red color for failure
NC='\033[0m'       #No color - to clear the color after

SINGER="./toolkit/singer"
STITCH="../ocaml/stitch"
TESTS="./_tests/*"
NTESTS="./_ntests/*"
TARGETS="./_targets"
OUTPUTS="./_outputs"
BIN="./_bin"
LOG="./_log/`date +%h%d_%H%M%S`_test_log.txt"

TCOUNT=0
PASSCOUNT=0

#print whether we succeeded or failed the test
function echoResult {

  if [ $1 -eq 0 ]; then
    PASSCOUNT=$((PASSCOUNT + 1))
    echo "${SUCC}TEST SUCCESSFUL!${NC}"
    echo "TEST SCCESSFUL!" >> $LOG
  else
    echo "${FAIL}TEST FAILED!${NC}"
    echo "TEST FAILED!" >> $LOG
  fi
}

function checkComp {

  if [ $1 -eq 0 ]; then
    echo "${SUCC}COMPILE SUCCESSFUL!${NC}"
    echo "COMPILE SUCCESSFUL!" >> $LOG
  else
    echo "${FAIL}COMPILE FAILED!${NC}"
    echo "COMPILE FAILED!" >> $LOG
    break
  fi
}

function checkNComp {

  echo "${SUCC}COMPILE FAILED!${NC}"
  echo "COMPILE FAILED!" >> $LOG
  break
}
```

```
55
56 ######################
57 # SCRIPT STARTS HERE #
58 ######################
59
60 #Make the compiler if it isn't already made
61 clear
62 echo "Making the compiler..."
63 cd ../ocaml
64 make all > /dev/null
65 cd ../testing
66
67 echo "*****************" 2>&1 | tee -a $LOG
68 echo "* Positive Tests *" 2>&1 | tee -a $LOG
69 echo "*****************" 2>&1 | tee -a $LOG
70
71
72 for test in $TESTS
73 do
74   TCOUNT=$((TCOUNT + 1))
75   echo "Starting Test $test" 2>&1 | tee -a $LOG
76   echo "====================" 2>&1 | tee -a $LOG
77   ROOT=`basename $test | cut -d'.' -f1`
78   $SINGER $test
79   checkComp $?
80   mv ./\_tests/$ROOT.stch.c ./\_targets
81   mv ./$ROOT $BIN
82   $BIN/$ROOT > $OUTPUTS/$ROOT\_gen.txt 2>&1
83   echo "\nDIFFing Output"  2>&1 | tee -a $LOG
84   echo "================"  2>&1 | tee -a $LOG
85   diff -w $OUTPUTS/$ROOT\_gen.txt $OUTPUTS/$ROOT\_out.txt
86   echoResult $?
87   echo "\n\n" 2>&1 | tee -a $LOG
88 done
89
90 echo "*****************" 2>&1 | tee -a $LOG
91 echo "* Negative Tests *" 2>&1 | tee -a $LOG
92 echo "*****************" 2>&1 | tee -a $LOG
93
94 trap checkNComp ERR
95
96 for test in $NTESTS
97 do
98   TCOUNT=$((TCOUNT + 1))
99   echo "Starting Negative Test $test" 2>&1 | tee -a $LOG
100  echo "============================" 2>&1 | tee -a $LOG
101  $STITCH $test 2> /dev/null || true
102  if [[ -e $test.c ]]; then
103    echo "${FAIL}TEST FAILED!${NC}"
104    echo "TEST FAILED!" >> $LOG
105    rm $test.c
106  else
107    PASSCOUNT=$((PASSCOUNT + 1))
108    echo "${SUCC}TEST SUCCESSFUL!${NC}"
109    echo "TEST SCCESSFUL!" >> $LOG
110  fi
111  echo "\n\n" 2>&1 | tee -a $LOG
112 done
```

```
113
114 echo Passed $PASSCOUNT / $TCOUNT tests 2>&1 | tee -a $LOG
115 echo "\n\n" 2>&1 | tee -a $LOG
116
117 cd $OUTPUTS
118 rm *_gen.txt
119 cd ../$TARGETS
120 rm *.c
121 cd ../$BIN
122 rm *
```

singer

```
1 #!/bin/sh
2 #Stitch complier toolchain
3 #Author: Dan Cole
4
5 FILENAME=`basename $1 | cut -d'.' -f1`
6
7 echo "----Stitch Compiler Toolchain-----"
8 ../ocaml/stitch $1
9 gcc -w ./_tests/$FILENAME.stch.c -I../runtime -L../runtime ../runtime/
    libstch_headers.a -o $FILENAME
```

Negative Tests

arith3.stch

```
1  //can't add chars to ints
2
3  int main()
4  {
5    int a = 0;
6    a = a + 'a';
7    print(a);
8
9    return 0;
10 }
11
12 /*Fatal error: exception Stch_semantic.Error("Incompatable data types for
       binop")*/
```

array2.stch

```
1  int main(){
2
3    int a[2*2] = {0,1,2,3};
4
5    int i = 0;
6
7    for(i = 0; i < 4; i = i + 1) {
8
9      print(a[i]);
10
11   }
12
13 }
```

array3.stch

```
/*wrong type in array initialization*/

int main()
{

  int a[3] = {1,0.5,2};

  return 0;
}

/*Fatal error: exception Stch_semantic.Error("Cannot initialize array with a
    variable")*/
```

array4.stch

```
1  /*initializing with 2D array on a 1D declared array fails.*/
2
3  int main()
4  {
5    int a[4] = {1,2,{3,4},5};
6
7    return 0;
8  }
9
10 /*Fatal error: exception Parsing.Parse_error*/
```

arrayinit1.stch

```
1  int main() {
2
3    int x = 5;
4
5    int a[x] = {1,2,3,4,5};
6
7    int i = 0;
8    for(i = 0; i < 5; i = i + 1) {
9      print(a[i]);
10   }
11
12   return 0;
13 }
```

arrayinit2.stch

```
1  int main() {
2
3    int i = 0;
4
5    int a[3] = {0,2,3,4,5};
6
7    for(i = 0; i < 3; i = i + 1) {
8
9      print(a[i]);
10   }
11
12   return 0;
13
14 }
```

char1.stch

```
int main(){

  char a = "hello";

  print(a);

  return 0;
}
```

comment2.stch

```
int main()
{
  print("one");
  //print("two");
  print("three");
  /*
    print("four");
    print("five");

  print("six");
}
```

comment4.stch

```
int main()
{
  print("one");
  //print("two");
  print("three");
  /*
    print("four");
    print("five");
  /*
  print("six");
}
```

error.stch

```
1 int main() {
2   void y;
3
4   error(y);
5
6   return 0;
7 }
```

exit2.stch

```
1 int main()
2 {
3   print("this should print"); //should not actually print
4   exit("bye!");
5   print("this should not");
6 }
```

file1.stch

```
int main()
{
  int a;
  char c[5000];
  a = open("file1.stch");
  read(a, c);
  print("success");
}
```

float1.stch

```
1  //floating point with multiple decimals.
2
3  int main()
4  {
5    float a = 1.2;
6    print(a);
7    float b = 1.23.4; //causes parsing error
8    print(b);
9
10   return 0;
11 }
12
13 /*Fatal error: exception Parsing.Parse_error*/
```

func1.stch

```
int main() {
  int b = 9;

  int func(int a){
    print(a);
  }

  func(b);

  return 0;
}
```

func2.stch

```
1  //error because there is no return type associated with foo().
2
3  foo(int a, int b)
4  {
5    print(a+b);
6  }
7
8  int main()
9  {
10   int a = 2;
11   int b = 3;
12   foo(a, b);
13
14   return 0;
15 }
16
17 /*Fatal error: exception Parsing.Parse_error*/
```

globalvar1.stch

```
1  /*global variables are not supported*/
2
3  int b = 1;
4
5  int main()
6  {
7    int a = 5;
8    print(a);
9
10   return 0;
11 }
12
13 /*Fatal error: exception Parsing.Parse_error*/
```

if1.stch

```
int main() {

  if(int i == 0){
    print("hello");
  }

  return 0;
}
```

if2.stch

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

  if(i == 1){
    print("1");
  }
  else{
    print("2");
  }
  else {
    print("3");
  }

  return 0;
}
```

matrixinit.stch

```
int main() {

  int i = 0;
  int j = 0;

  int b[2][2] = { {1, 2}, {2, 3, 4} };

  for(i = 0; i < 2; i = i + 1) {

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

      print(a[i][j]);

    }
  }

  return 0;
}
```

matrixinit2.stch

```
1 int main()
2 {
3   float a[2][2] = {{1.5, 2}, {3.5, 4.5}};
4
5   return 0;
6 }
```

negate2.stch

```
1 //negate with float data type, should not pass
2
3 int main()
4 {
5   int a;
6   a = !0.7;
7   print(a);
8 }
9
10 /*Fatal error: exception Stch_semantic.Error("Type mismatch on variable
     assignment a
11 Expected: int Got: float")*/
```

negate3.stch

```
1  //negate with wrong data type
2
3  int main()
4  {
5    int b;
6    b = !'c';
7    print(b);
8  }
9
10 /*Fatal error: exception Stch_semantic.Error("Cannot negate type char")*/
```

print.stch

```
1 int main() {
2   void y;
3
4   print(y);
5
6   return 0;
7 }
```

sem1.stch

```
int main() {
  int x = 2;
  float y = 3.4;

  print(x);
  print(y);

  if (x != y) {
    printf("false\n");
  }

  return 0;
}
```

sem3.stch

```
int main() {

  char a = 'A';

  print(a);

  int b = a;

  return b;
}
```

stitch1.stch

```
int main() {

  int arr[3] = {0,1,2};


  stitch i from 0 to 3 by 1: {
    arr[i] = 0;
  }
  return 0;
}
```

stitch4.stch

```
int main() {

  int i = 0;
  int test = 6;

  stitch i from 0 to 4 by 1:
    print(foo);

  return 0;

}
```

unfunc.stch

```
int main() {
  int a = 10;
  int b = 20;
  int c  = gcd(a,b);

  return c;
}
```

vardecl1.stch

```
/*identifiers cannot start with _ or a number*/

int main()
{
  int _a = 0;
  print(_a);
}

/*Fatal error: exception Failure("illegal character _")*/
```

void1.stch

```
1 int main() {
2
3   void a = "hello";
4
5   return 0;
6 }
```

Positive Tests

accum1.stch

```
int main() {

  int i = 0;
  int_ap dot = 0;

  int a[4] = {2,3,4,5};
  int b[4] = {4,3,4,3};


  stitch i from 0 to 4 by 1:{
    dot = a[i] * b[i];
  }

  print(dot);

  return 0;

}
```

arith1.stch

```
1 int main()
2 {
3   int a;
4   a = 39 + 3 + 10 + 42;
5   print(a);
6   return 0;
7 }
```

arith2.stch

```
1  //arith2.stch
2
3  int main()
4  {
5    int a = -5;
6    print(a);
7    int b = 5 * (8 + 3);
8    print(b);
9
10   return 0;
11 }
```

array1.stch

```
1  int main(){
2
3    int a[4] = {0,1,2,3};
4
5    int i = 0;
6
7    for(i = 0; i < 4; i = i + 1) {
8
9      print(a[i]);
10
11   }
12   return 0;
13 }
```

arrayassign.stch

```
1  int main(){
2
3    int a[4];
4
5    int i = 0;
6
7    for(i = 0; i < 4; i = i + 1) {
8
9      a[i] = i;
10
11   }
12
13   for(i = 0; i < 4; i = i + 1) {
14
15     print(a[i]);
16   }
17
18   return 0;
19 }
```

break1.stch

```
1  int main()
2  {
3    int a = 5;
4
5    while ( a > 1) {
6      print(a);
7      a = a - 1;
8      if (a == 3)
9        break;
10   }
11
12   print("passed while loop with break");
13
14   return 0;
15 }
```

collatz.stch

```
1  //Collatz Function
2
3
4  int c(int a) {
5
6    if(a%2){
7      return 3 * a + 1;
8    }
9    return a/2;
10 }
11
12 int main() {
13
14   int x;
15   x = 42;
16
17   while(x != 1){
18     x = c(x);
19     print(x);
20   }
21
22   return 0;
23 }
```

collatz2.stch

```
1  //Collatz Function
2
3  int c(int a) {
4
5    if(a%2){
6      return 3 * a + 1;
7    }
8    return a/2;
9  }
10
11 int main() {
12
13   int x;
14   x = 7859;
15
16   while(x != 1){
17     x = c(x);
18     print(x);
19   }
20
21   return 0;
22 }
```

comment1.stch

```
int main()
{
  print("one");
  //print("two");
  print("three");
  /*
    print("four");
    print("five");
  */
  print("six");

  return 0;
}
```

comment3.stch

```
int main()
{
  print("one");
  //print("two");
  print("three");
  /*
    print("four");
    //print("four point five");
    print("semis are for jive turkeys")
    print("five");
  */
  print("six");

  return 0;
}
```

escape.stch

```
int main() {

  char e = '\n';
  print(e);

  return 0;
}
```

exit1.stch

```
1 int main()
2 {
3   int x = 1;
4   print("this should print");
5   exit(x);
6   print("this should not");
7
8   return 0;
9 }
```

file1.stch

```
int main()
{
  FILE a;
  char c[5000];
  a = open_r("file1.stch");
  read(a, c);
  print("success");

  return 0;
}
```

file2.stch

```
int main()
{
  FILE a;
  char c[13] = {'h', 'e', 'l', 'l', 'o', ',',' ','w','o','r','l','d','!'};
  a = open_w("./_outputs/file2a_gen.txt");
  write(a, c);
  print("success");

  return 0;
}
```

for1.stch

```
/*variable declaration inside for loops work*/

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

  i = 0;
  while(i < 5)
  {
    int b = 2;
    print(b);
    b = b + 1;
    i = i + 1;
  }

  return 0;
}
```

func1.stch

```
1  //Calling a function from another function
2
3
4  void p(int a) {
5
6    print(a);
7
8  }
9
10 int main() {
11
12   int x;
13   x = 6;
14
15   p(x);
16
17   return 0;
18 }
```

func2.stch

```
1  void func1(int a){
2    print(a);
3  }
4
5  void func2(int b){
6    print(b);
7  }
8
9  int main() {
10   int x = 1;
11   int y = 2;
12   func1(x);
13   func2(y);
14
15   return 0;
16 }
```

func4.stch

```
1 int func(int x)
2 {
3   return x + 1;
4 }
5
6 int main()
7 {
8   int a = 0;
9   a = func(a=7);
10
11  print(a);
12
13  return 0;
14 }
```

func5.stch

```
1  float func(float a, float b, float c)
2  {
3    return a + b + c;
4  }
5
6  int main()
7  {
8    float a = 0.5;
9
10   a = func(1.0,2.0,3.0);
11   print(a);
12
13   return 0;
14 }
```

gcd.stch

```
int gcd(int a, int b) {
  while (a != b) {
    if (a > b) {
      a = a - b;
    }
    else {
      b = b - a;
    }
  }
  return a;
}

int main() {
  int x = 1;
  int y = 10;

  int z = gcd(x,y);

  print(z);

  return 0;
}
```

hello1.stch

```
1 int main(){
2
3   print("hello, world");
4
5   return 0;
6 }
```

hello2.stch

```
int main(){

  print("hello, ");
  error("world");

  return 0;
}
```

if1.stch

```
1 int main()
2 {
3   if (1) { print(42); }
4   print(17);
5
6   return 0;
7 }
```

if2.stch

```
1  int main()
2  {
3    int x = 17;
4    if (1) {
5      int x = 42;
6      print(x); }
7    print(x);
8
9    return 0;
10 }
```

if3.stch

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

  if(i == 1){
    print("1");
  }
  else{
    print("2");
    if(i == 0){
      print("3");
    }
  }

  return 0;
}
```

main.stch

```
int main(int x){

  int y = 10;
  print(y);
  return 0;
}
```

matmult.stch

```
int main() {

  int a[5][5] = { {1, 2, 3, 4, 5},
          {1, 2, 3, 4, 5},
          {1, 2, 3, 4, 5},
          {1, 2, 3, 4, 5},
          {1, 2, 3, 4, 5} };

  int b[5][5] = { {1, 1, 1, 1, 1},
          {2, 2, 2, 2, 2},
          {3, 3, 3, 3, 3},
          {4, 4, 4, 4, 4},
          {5, 5, 5, 5, 5} };

  int c[5][5];

  int i = 0;
  int j = 0;
  int k = 0;

  stitch i from 0 to 5 by 1: {

    for(j = 0; j < 5; j = j + 1) {

      for(k = 0; k < 5; k = k + 1) {

        c[i][j] = c[i][j] + a[i][k] * b[k][j];
      }
    }
  }

  for(j = 0; j < 5; j = j + 1) {

    for(k = 0; k < 5; k = k + 1) {

      print(c[j][k]);
    }
  }


  return 0;
}
```

matrix1.stch

```
int main() {

  int m[3][3];

  int i = 0;
  int j = 0;
  int k = 0;

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

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

      m[i][j] = k;
      k = k + 1;
    }
  }

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

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

      print(m[i][j]);
    }


  }

  return 0;
}
```

matrixinit.stch

```
int main() {

  int a[2][2] = { {1,2}, {3,4} };
  int i = 0;
  int j = 0;

  for(i = 0; i < 2; i = i + 1) {
    for(j = 0; j < 2; j = j + 1) {

      print(a[i][j]);
    }
  }

  return 0;
}
```

matrixstitch.stch

```
int main() {

  int i = 0;
  int test = 6;

  int a[6][6];
  int k = 0;
  int j = 0;

  for(k = 0; k < 6; k = k + 1) {
    for(j = 0; j < 6; j = j + 1) {
      a[k][j] = 0;
    }
  }

  stitch i from 0 to 6 by 1: {

    int j;
    for(j = 0; j < 6; j = j + 1) {
      a[i][j] = a[i][j] + 10;
    }
  }

    for(j = 0; j < 6; j = j + 1) {
      for(k = 0; k < 6; k = k + 1) {
        print(a[j][k]);
      }
    }

  return 0;

}
```

negate.stch

```
1  //negation test
2
3  int main()
4  {
5    int a = 0;
6    int b = !a;
7    print(b);
8
9    return 0;
10 }
```

ops1.stch

```
int main()
{
  print(1 + 2);
  print(1 - 2);
  print(1 * 2);
  print(100 / 2);
  print(4 % 2);
  print(99);
  print(1 == 2);
  print(1 == 1);
  print(99);
  print(1 != 2);
  print(1 != 1);
  print(99);
  print(1 < 2);
  print(2 < 1);
  print(99);
  print(1 <= 2);
  print(1 <= 1);
  print(2 <= 1);
  print(99);
  print(1 > 2);
  print(2 > 1);
  print(99);
  print(1 >= 2);
  print(1 >= 1);
  print(2 >= 1);

  return 0;
}
```

ops2.stch

```
1 int main()
2 {
3   int a = 2;
4   int b = 3;
5   if (a == 2 && b == 3)
6     print(a);
7   else
8     print(b);
9
10  if (a != 2 || b <4)
11    print(b);
12  else
13    print(a);
14
15  return 0;
16 }
```

sem2.stch

```
int main() {

  char z = 'z';

  print(z);

  char y = z;

  print(y);

  return 0;

}
```

stitch1.stch

```
int main () {


  int i = 0;
  int test = 6;

  stitch i from 0 to 4 by 1: {
    test = 7;
    print(test);
  }

  return 0;
}
```

stitch2.stch

```
int main() {

  int i = 0;
  int test = 6;

  stitch i from 0 to 4 by 1: {
    test = 7;
        print(test);
  }

  i = 0;

  stitch i from 0 to 4 by 1: {
      test = 9;
      print(test);
  }

  return 0;

}
```

stitch3.stch

```
int main() {

  int i = 0;
  int test = 6;
  test = 7;

  stitch i from 0 to 4 by 1:
    print(test);

  return 0;

}
```

stitch4.stch

```
int main() {

  int i = 0;
  int test;
  test = 6;

  stitch i from 0 to 4 by 1: {
    int test = 8;
    print(test);
    }

  return 0;

}
```

stitch5.stch

```
int main() {

  int i = 0;
  int test = 6;
  test = 8;
  int j = 0;

  for(j = 0; j < 4; j = j + 1){

    stitch i from 0 to 4 by 1:
      print(test);
  }

  return 0;

}
```

stitch6.stch

```
int main() {

  int i = 0;
  int test = 6;
  test = 7;

  {
    stitch i from 0 to 4 by 1:
      print(test);
  }

  return 0;

}
```

stitch7.stch

```
int main() {

  int i = 0;
  int test = 6;

  int a[10];
  int k = 0;

  for(k = 0; k < 10; k = k + 1) {
    a[k] = k;
  }

  stitch i from 0 to 10 by 1: {

    int j;
    j = 7;
    a[i] = a[i] + 1;
    }

    int j = 0;
    for(j = 0; j < 10; j = j + 1) {
      print(a[j]);
    }

  return 0;

}
```

Parser Tests

dec1.stch

```
1 /*Standard variable declaration*/
2
3 int main(){
4   int a;
5 }
```

dec2.stch

```
1 /* Variable dec followed by assignment*/
2
3 int main(){
4   int a;
5   a = 5;
6 }
```

dec3.stch

```
1 /* Variable dec together with assignment*/
2
3 int main(){
4   int a = 5;
5 }
```

dec4.stch

```
1 /*Integer by itself as a declaration*/
2
3 int main(){
4   5;
5 }
```

dec5.stch

```
1 /*Variable decs followed by assignments*/
2
3 int main() {
4   int a;
5   a = 2;
6   int b;
7   b = 7;
8   int c;
9 }
```

dec6.stch

```
1 /*String literal by itself*/
2
3 int main(){
4   "test";
5 }
```

dec7.stch

```
1 //Tests expr followed by parens
2
3 int main() {
4
5   a;(5 + 5);
6
7   return 0;
8 }
```

dec8.stch

```
// Testing arrays, both should work

int main() {

  int a[5 + 2];

  float f[];

  //this should also work
  a[4] = 7;

  int b[4] = {5, 4, 3, 2};

  return 0;

}
```

func1.stch

```
1 /*Function that returns 0, no args*/
2
3 int foo() {
4   return 0;
5 }
```

func2.stch

```
/*Multiple functions, one called from other*/

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

func3.stch

```
/* Variable declaration followed by a function call*/

void foo() { }

int main() {
  int a;
  a = 10;
  foo();
  return 0;
}
```

func4.stch

```
1 /* Variable declaration followed by function call, returning value */
2
3 int foo() {
4   return 5;
5 }
6 int main() {
7   int a;
8   a = foo();
9 }
```

func5.stch

```
1 /*Return variable from a function*/
2
3 int main () {
4   int a;
5   a = 5;
6   return a;
7 }
```

func6.stch

```
1  /*Return nothing from a function*/
2
3  void foo(){
4    return;
5  }
6
7  int main () {
8    foo();
9    return 0;
10 }
```

func7.stch

```
1 /*Testing access operation*/
2
3 int main() {
4
5   int a;
6
7   a.element;
8 }
```

loop1.stch

```
/*If else with equality*/

int main() {
  int a;
  a = 1;
  if (a == 1){
    return 1;
  }
  else {
    return 0;
  }
}
```

loop2.stch

```
/*If else conditionals*/

int main() {
  int a;
  int b;
  a = 1;
  if (a > 0){
    b = 0;
  }
  else {
    b = 1;
  }

  return b;
}
```

loop3.stch

```
/*For loop test*/

int main() {
  int a = 0;
  int b = 0;

  for (b; b < 5; b = b + 1){
    a = a + 1;
  }

  return a;
}
```

loop4.stch

```
/*Stitch Loop Test*/

int main() {

  int a;
  a = 5;

  int i;

  stitch i from 0 to 10 by 1 : {

    a = 7;

  }

  return 1;

}
```

loop5.stch

```
1  //While loop with conditional
2
3  int main() {
4
5    int a;
6    int b;
7    a = 1;
8    b = 5;
9
10   while(a < 4) {
11
12   a = b;
13
14   }
15
16   return 0;
17 }
```

loop6.stch

```
1  //Testing a for loop with argument 1 only
2
3  int main() {
4
5    int x = 0;
6
7    for(x;;) {
8
9    }
10 }
```

loop7.stch

```
1  //Testing a for loop with argument 2 only
2
3  int main() {
4
5    int x = 0;
6
7    for( ; x < 7; ) {
8
9    }
10 }
```

loop8.stch

```
1  //Testing a for loop with argument 3 only
2
3  int main() {
4
5    int x = 0;
6
7    for(;;x = x + 1) {
8
9    }
10 }
```

return1.stch

```
1  /*Return nothing from a function*/
2
3  void foo(){
4    return;
5  }
6
7  int main () {
8    foo();
9    return 0;
10 }
```

Runtime

Makefile

```
CC = gcc
CXX = g++

INCLUDES = -g -Wall #-I

CFLAGS = $(INCLUDES)
CXXFLAGS = $(INCLUDES)

LDFLAGS = -g #-L
LDLIBS =

stch_headers_LIB: stch_headers.o
  ar rc libstch_headers.a stch_headers.o
  ranlib libstch_headers.a

stch_headers.o:

.PHONY: clean
clean:
  rm -f *.o a.out core libstch_headers.a stch_headers.o

.PHONY: all
all: clean stch_headers_LIB
```

stch_headers.c

```c
/*
 * stch_headers.c
 * library of standard Stitch functions
 */

#include "stch_headers.h"

//open()
// int  stch_open(const char* source){
//  return fopen(source, "r+");
// }
//write()
// int  stch_write(const int fd, stch_array* source){
//  return write(source->data, source->length, 1, fd);
// }
//read()
// int  stch_read(const int fd, stch_array* dest){
//  return read(source->data, source->length, 1, fd);
// }

//lengthof()
int  stch_length(const stch_array* a){
  return a->length;
}
//cut()
void stch_cut(void* e){
  pthread_exit(e);
}
```

stch_headers.h

```c
/*
 * stch_headers.h
 * auto-included in ever c file written by the Stitch compiler
 */

#ifndef __STCH_HEADERS_H__
#define __STCH_HEADERS_H__

/*
************
* Includes *
************
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>

/*
***********
* Defines *
***********
*/
#define NUMTHREADS 4

/*
***********
* Structs *
***********
*/

//hold local variables to pass from the stitch loop into a thread
//need to figure this out...
struct stch_LocalVars{

    void      *vars;
    unsigned int  n;

};

// //range info passed into the thread
// struct stch_rangeInfo{

//     int  begin;
//     int  end;
//     int  stepSize;
```

```
47 //      int  cols;
48 //      struct   stchLocalVars *locals;
49 //      void *myvars;
50
51 // };
52
53 //array wrapper
54 typedef struct stch_array{
55
56   char       *name;
57   unsigned int  length;
58
59 } stch_array;
60
61 /*
62 ************************
63 * Function definitions *
64 ************************
65 */
66
67 //open()
68 int  stch_open(const char* source);
69 // //write()
70 int  stch_write(const int fd, stch_array* source);
71 // //read()
72 int  stch_read(const int fd, stch_array* dest);
73 // //lengthof()
74 int  stch_length(const stch_array* a);
75 // //cut()
76 void stch_cut(void* e);
77
78
79 #endif
```

Demo

image_contrast2.stch

```
/* Image Contrast */

int main(){

  int curve[256] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 3,
     3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10, 11, 11, 12, 13, 13, 14,
    15, 15, 16, 17, 18, 19, 20, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
    31, 32, 33, 34, 35, 36, 37, 38, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50,
    52, 53, 54, 55, 57, 58, 59, 61, 62, 64, 65, 66, 68, 69, 70, 72, 73, 75,
    76, 78, 79, 81, 82, 83, 85, 86, 88, 89, 91, 92, 94, 96, 97, 99, 100, 102,
     103, 105, 106, 108, 109, 111, 113, 114, 116, 117, 119, 120, 122, 124,
    125, 127, 128, 130, 131, 133, 135, 136, 138, 139, 141, 142, 144, 146,
    147, 149, 150, 152, 153, 155, 156, 158, 159, 161, 163, 164, 166, 167,
    169, 170, 172, 173, 174, 176, 177, 179, 180, 182, 183, 185, 186, 187,
    189, 190, 191, 193, 194, 196, 197, 198, 200, 201, 202, 203, 205, 206,
    207, 208, 210, 211, 212, 213, 214, 215, 217, 218, 219, 220, 221, 222,
    223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 235,
    236, 237, 238, 239, 240, 240, 241, 242, 242, 243, 244, 244, 245, 246,
    246, 247, 247, 248, 248, 249, 249, 250, 250, 251, 251, 252, 252, 252,
    253, 253, 253, 254, 254, 254, 254, 255, 255, 255, 255, 255, 255, 255,
    255, 255, 255, 255 };


  FILE inFile = open_r("img.bmp");
  FILE outFile = open_w("out_hc.bmp");
  char buffer[98592];
  read(inFile, buffer);

  int i = 0;
  //BMP header offset
  stitch i from 55 to 98592 by 1:{
    int tmp = 0;
    tmp = buffer[i];

    if (tmp < 0) {
      tmp = tmp + 256;
    }

    buffer[i] = curve[tmp];
  }

  write(outFile, buffer);

  return 0;
}
```

image_invert2.stch

```
/* Image Invert */

int main(){

  int curve[256] = { 255, 254, 253, 252, 251, 250, 249, 248, 247, 246, 245,
    244, 243, 242, 241, 240, 239, 238, 237, 236, 235, 234, 233, 232, 231,
    230, 229, 228, 227, 226, 225, 224, 223, 222, 221, 220, 219, 218, 217,
    216, 215, 214, 213, 212, 211, 210, 209, 208, 207, 206, 205, 204, 203,
    202, 201, 200, 199, 198, 197, 196, 195, 194, 193, 192, 191, 190, 189,
    188, 187, 186, 185, 184, 183, 182, 181, 180, 179, 178, 177, 176, 175,
    174, 173, 172, 171, 170, 169, 168, 167, 166, 165, 164, 163, 162, 161,
    160, 159, 158, 157, 156, 155, 154, 153, 152, 151, 150, 149, 148, 147,
    146, 145, 144, 143, 142, 141, 140, 139, 138, 137, 136, 135, 134, 133,
    132, 131, 130, 129, 128, 127, 126, 125, 124, 123, 122, 121, 120, 119,
    118, 117, 116, 115, 114, 113, 112, 111, 110, 109, 108, 107, 106, 105,
    104, 103, 102, 101, 100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88,
    87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70,
    69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52,
    51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34,
    33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16,
    15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 };


  FILE inFile = open_r("img.bmp");
  FILE outFile = open_w("out_invert.bmp");
  char buffer[98592];
  read(inFile, buffer);

  int i = 0;
  //BMP header offset
  stitch i from 55 to 98592 by 1:{
    int tmp = 0;
    tmp = buffer[i];

    if (tmp < 0) {
      tmp = tmp + 256;
    }

    buffer[i] = curve[tmp];
  }

  write(outFile, buffer);

  return 0;
}
```