Variable Names:
All variables should be declared and initialized at the start of the the main function.
Camelcase should be used for non_constant variables. (varName)
All caps with underscore spacing should be used for constant variables. (CONSTANT_VAR)
Temporary variables should use the prefix tmp. (tmpVar)
All variables should be declared on seperate lines.

All structs should use typedef.
All structs variables should begin with a capital.

Pointer variables should be declared with an asterix beside the type. (Char* a)

.h libraries using <> should be included before those using “”.

Function Names & Features:
Function names should be camelcase with the first letter capitalized. (FunctionName)
Void functions should contain an empty return statement.

Curly Braces:
Braces designating a code block for a function should be on a separate line from the function name, whereas braces designating a code block for loops, if statements, etc, should begin on the same line as the statement.

ie. function

```
int FunctionName(param)
{
        //code block
}
```

ie. other

```
if (boolean) {
        //code block
}
```

Loops and if statements should always be contained within braces, even if the contents is only a single line.

Else statements should begin on a seperate line the the previous if statements closing brace.

Indentation/WhiteSpace:
Statements should be indented under the statement to which they are logically subordinate.

Indentation should consist of 4 white spaces (not tabs).

A blank line should separate all major and minor code bodies.
ie. using a blank line to seperate variable declarations from if statements.

Buffers and parentheses should be used to increase code readability wherever possible.
ie.
In the case of: num=1+var1/4+var2
Instead show: num = 1 + (var1 / 4) + var2

Commenting:
Not all code needs to be commented, however all code deemed confusing or possibly confusing should be clarified.
Header comments for functions should use the block comment syntax.
Doxygen format should be used for these comments.
Header comments should only be included within the header files.
ie.

```
/*
 * comment
 */
FunctionName();
```

All comments used to to explain sections of code should use the block comment syntax and be located above the relevant code block.
ie.

```
/*
 * explanation of loop
 */
for ( i = 0; i < num; ++i) {
        //code block
}
```

Comments used to specify/clarify small details of a single line of code that may be confusing, should be on the same line as that code and use the single line syntax. (//)
Details like these should be as brief as possible. (hopefully 1 - 3 words)
ie.

```
confusing line of code; //specific detail clarified
```

Comments used to explain the purpose of variables during declaration should be located on the same line and use the single line syntax. (//)

ie.

```
int var = 5;            //var used for etc
```

File Documentation:
.H has function declarations, .C has implemented functions

Do Nots:

1. goTo statements
2. break statements

Memory:
Valgrind should be used whenever possible/necessary.

Compilation:
Programmers should compile with the following flags:

- -Wall
- -std=c99
- -g
- -pedantic

Code should be compiled using a makefile.
Preprocessor DDebug will be used.

File Locations and Requirements
Program files should be organized using the following directory format contained within a top level directory.
The top level directory should contain the following folders.

1. assets: will include any data files.
2. bin: will include all programs and executables. (.o)
3. includes: will include all header files. (.h)
4. src: will include all source files. (.c)
5. docs: will include all project documentation, with the exception of the README.

The top level directory will also include the makefile and the README file.