CODING STYLE GUIDELINES
ECE 2534 – Fall 2014
(Adapted from those of Prof. Jill Seaman of Texas State University.)

Header comments (a file documentation block) should be at the top of each file that you create. It should contain information similar to that given in the example below.

```
//////////////////////////////////////////////////
// ECE 2534, Homework 2, your name
//
// File name:   hw2.c
// Description: . . .
//
// Date:        11/11/2014
//
```

`#include` directives must follow header comments, before the rest of the program.

Comments for variables and functions:
Variable definitions should be commented.

Each nontrivial function should have a comment block at the beginning to describe its purpose, its parameters, and what the function returns.

If a function body contains more than about five statements, there should be comments to describe the various sections of code in the function body.

White space:
--use blank lines to separate sections of code
--try to use consistent indentation amounts (e.g., 4 spaces) throughout your program
--be consistent with placement of { }, ( ), etc.

Line length of source code should be no longer than 80 characters.

Variable names:
--should be meaningful, in general
--loop index names can be simple (`i`, `j`, `k`, etc.), although using meaningful names here can often help with debugging (`arrayIndex`, `loopidx`, etc.)
--The initial letter should be lowercase, and "camel-casing" is desirable (e.g., `portInputValue`).

Named constants:
--use for most numeric literals, including array sizes
--all capitals with underscores (e.g., MAX_INPUTS, STATE_SALES_TAX)
--should occur at top of function, or (only if necessary) as a global at the top of a file
--it is generally a good idea to use a named constant instead of a `#define`

--example: `const int MAX_SIZE = 100;`

Coding:

--a C function should be short and simple, as a general rule. Your main() function can essentially be a collection of function calls.

--avoid global variables. It is usually much better to use variables that are locally scoped.

--do not use `goto` statements. In this class there are no circumstances that require a `goto`, and in real life such occasions are very, very rare. Use smart conditionals and loop structures instead.

--use `{ }` instead of `;` after a `while` statement, even if the code block inside the braces is empty. This will save you hours of debugging time, because those semicolons are easy to overlook.

--header files (*.h) should have “include guards”, as a general rule, to prevent the file from being included multiple times. Example:

```
#ifndef FILENAME_H
#define FILENAME_H
...
#endif
```

--do not use `#include` *file*`.c`