ECE 2220 Programming Guide Fall 2016

Coding Standards

1. Include a comment block at the top of each file. The header comment must contain your name, Clemson login name, course number, semester, and assignment number. In addition the header comment must contain a section called "Purpose" in which you describe the purpose of the code in that file. *Be brief and informative.* Next, include a section called "Assumptions" and state the assumptions you make regarding the problem to be solved, the kinds of inputs you program will accept, and any deviations from the assignment requirements. Also, list any known bugs. Specific assumptions regarding particular functions should be expressed in comment blocks above the relevant functions.

2. The file with the `main()` function must be named `labX.c` where `X` is the assignment number.

3. You must include a brief comment block at the beginning of each function and structure definition stating its purpose. For a function, you must define the purpose of each input and the values that are valid for the input(s). You must also define the results that are returned, and provide comments about any data structures that are changed by the function

4. Avoid gratuitous comments. Use inline comments sparingly but whenever necessary to explain critical portions of the code. Comments should be used to explain any non-obvious details. Comments should indicate what is happening, not how it is being done. For short functions, the block comment preceding the function is often sufficient. Do not write comments that restate the code (e.g. // an if statement). Observe the rule of thumb that if you have more lines of comments than lines of code, you have too many comments.

5. Write self-documenting code. Choose meaningful names for all variables, parameters, and functions. Use complete words instead of abbreviations where practical. If a constant is needed more than one time, use a named constant. The size of an array should **always** be defined using a named constant.

6. Global variables cannot be used. If you can give a **very** convincing argument why the variable must be global, then check with me. Similarly, goto statements cannot be used.

7. Use common sense. Avoid excessive statements per line, or overly long lines. Use a consistent indentation style. This is just a guide, and your primary concern should be in making sure that others can read and understand the text of your program.

Grading will be based on three components

You must compile with `gcc` under a recent version of Ubuntu and use the `-Wall` flag. Code that does not compile or compiles but does not run will not be accepted. Points will be deducted for code that compiles but has compiler generated warning messages unless you document a reason why your code cannot be modified to eliminate the warning messages.

Code design (25%) We will evaluate organization of the interfaces in your program, how your program is broken into pieces, how the pieces communicate, what data structures and algorithms you decide to use for the pieces.

Documentation (25%) Equally important to good code design is ensuring that your code can be read and possibly modified by other people. We will grade on how well you have adapted a clear and meaningful style.

Correctness (50%) We will compile and run your code and test it against our set of test cases. We will also read your code to verify that it follows the requirements specified in the assignment.

Code that does not compile or that crashes before running any test cases will not be accepted for grading.