CS 1714 - Introduction to Programming II

Coding Style Guidelines

All students are required to follow these coding style guidelines for all assignments. The grading rubric for programming assignments will have a graded component for coding style.

File Names

Generally, file names should be lower-cased with no spaces and descriptive of what the file contains. Related source and header files should share the same file name, but have different extensions. Lab assignments and projects may have specific file naming requirements.

Identifiers

Identifier names for variables, functions, and structs must be descriptive. Use camel-casing for identifier names for variables, functions, and structs. Function and variable names are camel-cased beginning with a lower-case letter whereas structs begin with a capital letter.

Variable Names

For variable names, identifiers should describe the information they hold.

```
points
width
characterName
numberOfTurns
```

Function Names

In the case of function, the identifier should start with an action verb describing what the function does.

```
calculateTotal()
getValues()
setValues()
addNumbersTogether()
printLyrics()
```

Struct Names

Struct names are camel-cased beginning with an upper-case letter. Struct names should also be descriptive of the objecgt they represent.

```
Student
StudentRecord
BookInformation
```

Constants

Constants are named with all upper-case letters. When constant names contain more than one word, separate words with an underscore. Remember to use the `#define` preprocessor statement in the header file to define constants.

```
PI
NUMBER_OF_US_STATES
NUMBER_OF_SIDES_IN_A_TRIANGLE
```

Curly Braces and Indentation

Curly braces used in code blocks to contain one or more statements will follow this rule. They are used for control structures, loops, functions, and structs.

Each curly brace, whether it is opening or closing, must be on its own line. The opening curly brace must be vertically aligned to the beginning of its heading (e.g., if-statement, function signature, for-statement, etc.) The closing curly brace must be vertically aligned with its corresponding opening curly brace.

Any statements within the curly braces should be indented: tabbed once.

```
void calculateValues( int number )
{
      if( number > 10 )
      {
            printf( "Hello" );
      }
      else
      {
            for( int index = 0; index < 2; index++ )
            {
                  printf( number );
            }
      }
}
```

Curly braces used for arrays may ignore these rules.

The traditional C style would have the opening curly brace on the same line as the initial statement. This style is acceptable, but makes it harder to debug.

Operators

There should be exactly one space on either side of an operator. This coding style refers mostly to arithmetic and Boolean expressions.

```
int sum = 5 + 8;
int sum = ( ( 5 + 7 ) * 6 – 4 ) * 8;
bool condition = flagOne != 5 && flagTwo < 12;
```

The only exception is with the opening parenthesis for a function, if-statement, or loop. In this case, there is no space to the left of the opening parenthesis—it will come right after the function name, if keyword, or loop keyword.

```
for( int index = 0; index < 5; index = index + 1 )

if( value < 5 )

void printValues( int points )

printValues( 35 );
```

Comments

Any code should be documented with comments. This section describes all the comments you should have for code submitted in this class. Lab assignments should have comments whenever possible, but are **required** for projects.

File Comments

Use comments to describe your source code. All .c and .h files must have these comments at the top.

```
/*
      name-of-file.c
      Assignment N
      Firstname Lastname

      Description of what this program is and how this file contributes
      to the program.
*/
```

Function Comments in .c files

Every function in your source code must have this immediately above the header. You do not need to have a parameter listing if your function has no parameters. If the return type is void, you can say that it returns nothing.

```
/*
      Function: functionName
      -------------------
      Brief description of what this function does

            parameter1identifier: description of the first parameter
            parameter2identifier: description of the second parameter

      Returns: Description of what the return value is
 */
```

Function Comments .h Files

The function declarations in the .h file require comments as often, that may be the only file a programmer may see. You only need to copy the full function comment from the .c file and put it above its respective function prototype. Since there may be no parameter names in the function declaration, you can list them as parameter 1, parameter 2, parameter 3, and so on.

Other Comments

Be sure to use comments throughout your code, especially when you have a lengthy piece of code or it may not be apparent as to the average reader. Remember comments are used to help you as well as others understand your code. There is no need to go overboard with comments.