Programming Style Guidelines

Programming style is a way of using statements, identifiers, comments, and whitespace to create programs that are easy to read, understand, debug, and maintain.

You are **required** to follow the style guidelines given below in this course.

General Style Guidelines:

- Provide clear prompts whenever you expect the program user to enter anything.
- Do not just display resulting value. Annotate all outputs with messages or clear labels. Include units if possible.
- Provide appropriate comments in your source code.
  - Document the purpose of the program with a comment at the top of the source file. Describe the inputs and outputs produced.
  - Explain sections of the program including complex code segments and complex calculations.
  - For each function in your program (except **`main`**), put a comment above the function definition explaining what the function does, its parameters, and return value (if the function returns something). Be sure to include data types where applicable.
- Place no more than one statement on a line.
- Spread long statements and comments over multiple lines so that readers do not have to scroll to view your code on a computer monitor.
- Use whitespace characters (blank lines, spaces, and tabs) to visually arrange the program in a way that is both pleasing and readable.
  - Separate distinct program components with a blank line. Program components include multiple sequential statements that are similar, for example multiple variable definitions which span multiple lines.
    - Program definitions (variables, constants, etc.)
    - Function definitions.
    - Decision constructs (**`if`** statements, **`if`**/**`else`** statements, and **`switch`** statements)
    - Loops (**`while`** loops, **`do-while`** loops, and **`for`** loops)
  - Indent the statements in a code block (code inside of a set of curly braces) one level from the surrounding code.
  - Put a space on both the left and right sides of an operator (when possible) to make the operator stand out visually.
- Use meaningful identifiers. Give your variables, constants, and functions meaningful names.
- Follow naming conventions when choosing the names of variables, constants, functions, etc. For example, variable names should begin with a lowercase letter and constants should be in all caps.
- Do not use global variables. Define all variables locally – inside the body of some function.
- Do not use the **`goto`** statement.
- Do not use the **`continue`** statement in this course.
- Do not use the **`exit`** statement in this course.

Decision Style:

- Begin the conditionally executed code block on the line below the **`if`** or **`else`** in a decision structure.
- Put the opening curly brace under the **`i`** of **`if`** or the **`e`** of **`else`**. Then on a new line indent the conditionally executed statement(s) one level from the key words **`if`** or **`else`**. Put the closing curly brace on a line by itself such that it aligns vertically with the opening curly brace.
- Indent the statements in a **`switch`** block.
- Put individual **`case`** labels on separate lines.
- Begin the statements under a **`case`** label on the line after the label and indent them one level.
- The **`else`** corresponding to an **`if`** should be at the same level of indentation as the **`if`** in all decision statements except when you are coding a multi-way decision using an **`if`**/**`else if`** structure.
- Do not use a **`return`** statement or call the ***exit*** function to avoid coding the **`else`** part of a decision. It is desirable to have only one exit point from a program.

Loop Style:

- Begin the body of a **`while`** or **`for`** loop on the line below the loop header.
- Begin the body of a **`do-while`** loop on the line below the key word **`do`**.
- Indent the statements in the body of a loop one level from the key word that marks the beginning of the loop.
- Write your loops so that it is clear from the loop continuation expression when the loop iterates and when it terminates.
  - Do not put statements in the body of a **`for`** loop that change the value of the variable(s) used to control the iterations of the loop.
  - Do not use a call to the ***exit*** function to terminate the iterations of a loop.
  - Do not use a **`return`** statement to exit a loop.
  - Do not use a **`break`** to exit a loop in this course.
  - Do not use **`continue`** in this course.

Function Style:

- Document your functions with a preceding comment, as described on the previous page.
- If the function inputs data from the program's user or outputs data to the console, then provide comments describing those inputs and outputs.
- Give your functions meaningful names.
- Indent the statements in the body of a function one level from the header.
- Define variables locally (inside functions). Send data items to other functions using arguments/parameters.
- Do not use multiple **`return`** statements in a function. It is desirable to have a single exit point from a function. The **`return`** statement, if there is one must be the last statement of the function body.

This document is not guaranteed to be complete. Make sure you follow the formatting conventions described and shown in the lectures.