Jeremy Prater

CS 161 – Assignment 7 Comparisons

Comparison 1: Candace.findMode.cpp

The first thing that struck me as odd was the global variables for result, the cons tint Num=10000, and the arrayMode[Num]. std::vector is a dynamic allocation entity, by declaring a global const variable which limits the capacity of the program, but not the system is a design flaw.

The variable arrayMode is referenced inside of the findMode(...) function. This practice should be avoided unless there is an absolute need to reference a global variable inside of a function (c embedded development and other resource limited applications can benefit from this).

The logic inside of findMode was hard to follow. The comments and variable names were difficult to understand. Also findMode used std::cout to print debug information, but the function was supposed to return the result vector with no other information provided, so that is a defect.

Comparison 2: Tira.findMode.cpp

This example started of excellent with the usage of ‘using namespace std’ to minimize namespace referencing for commonly used namespaces. The function prototype for findMode was clearly defined.

This example uses two loops, one to determine the maximum frequency, and one to add elements to the output vector.

The output vector is sorted before returning.

This example has good comments and variables are named more appropriately that the first example.

Comparison 3: Patrick.findMode.cpp

This example also ‘using namespace std’ to help reduce qualification of commonly used namespaces.

This example uses a nested loop which reduces code complexity as there are fewer logic paths, but could increase run time because once

all the final modes are added to the result vector, the program may waste time iterating n*n loops.

This example has good comments and easy to understand variable names.

Improvements:

Comparing this to my own code, I can see that variable names and comments are helpful. When I am the only one reading/debugging the code, I know what the variable names mean, but adding comments or better worded variable names help others easily read and understand the code.

Also, eliminating excessive looping and use of global variables would help performance of larger, more computationally expensive programs.