First Annual Programming Efficiency Conference and Challenge

{**aPECc**}

Code Submission for programming challenge is due (deadline) on **March 27, 2016**

Email your cpp and doc file to [Daniel.Creider@tamuc.edu](mailto:Daniel.Creider@tamuc.edu)

Top prize for one graduate student and one undergraduate student

2TB portable HD

*Award based on correct results and fastest execution time*

**Code to Write**

Write a **void function** that will find the modes(s) of a data set and store the actual mode values in an array passed to the function. Submit only the function you wrote; do not submit your complete program including the main module you wrote to test your function. There will be at least one mode in the data set but there could also be many modes which occur with the same frequency. The first line of your function definition must be the following. Do not change the order or type of the parameters or your function will not match the call statement in the main module that has been written to test your code.

void mode(long data[], long size, long &num\_modes, long modeNums[], long &maxFrequency)

The data that will be used to validate your code will consist of at least 30 million ***random*** values of type long. The main code has been written to test your function and will output the results to determine if your code obtained the correct results. **Do not input or output any values/data in the function that you submit**. The main program will pass the data array and the number of values stored in the data array to your function. Your function will compute the number of mode(s) in the data set and assign that value to the parameter, *num\_modes* (this represents the number of modes you found which also is the number of mode values stored in the *modeNums* array); your program will store the actual value of the mode(s) in successive elements of the *modeNums* array; and your program will compute the number of times the mode(s) occurred in the data set and assign that value to the parameter, *maxFrequency*. All modes must occur with the same maximum frequency.

You are permitted to use any algorithm of your choice. There are no restrictions on the manipulation of the data set. The main program will pass a copy of the original data set to your function. If you think you need more than one function to solve the problem you will have to put the prototype of a second function immediately after the first line of the function that the main module will call. The main module will only call 1 function. The code you write would look like the following if you write more than one function.

void mode(long data[], long size, long &num\_modes, long modeNums[], long &maxFrequency)

{

void secondFunction(parameter list); //prototype for second function if needed

// remainder of code in the mode function

}

void secondFunction(parameter list) // possible second function if needed

{

// code in the second function

}

Test your code with a data set for which you know the correct results before you try to use the TestData file. To test your code with a test data set of 30 million values you will need to include the following code in your main module. It is possible that you will not be able to create a compile time array that has 30 million elements so the code below will create a pointer and dynamically allocate an array of type long of the number of elements corresponding to the number of data values stored in the array. The test data set will be a binary file to minimize the size of the file. The following code is needed to dynamically allocate memory and read a binary file which is in a different file format than a text file.

#include <fstream> // header file needed to read a file

long \*data, size; // pointer needed to dynamically allocate memory

// the name of the file containing the test data is “TestData.bin”

ifstream binaryin("TestData.bin", ios::binary); // statement to open the binary file

binaryin.read(reinterpret\_cast<char \*>(&size), 4); // statement to read the size of the data array to allocate

data = new(nothrow)long[size]; // statement to dynamically allocate the data array to have *size* elements

if(!data) // this statement will test the pointer to determine if memory was dynamically allocated

{ cout<<"Memory allocation error for data array, program will terminate\n";

system("pause"); // this is a windows command to pause the program

exit(0); } // this statement will terminate the program if memory was not allocated

// statement to read the data/values in the file and store the numbers in the array *data*.

binaryin.read(reinterpret\_cast<char \*>(data), size\*sizeof(long));

The data will now be in the array and the rest of your code will go after this statement.

After memory is allocated and the values are stored in the array you can use the pointer *data* as you would a compile time array called *data*. Subscripts are permitted to be used with the variable *data* but not required. Such as - cout<<data[i]<<endl; You can use pointers or subscripts in your code.

You will need to time your code and report the results when you submit your code for the programming challenge. You can time your code with a built in function in C++ and the code to do this follows. The following variables need to be declared.

long double time1; // variables used to time the code

clock\_t start,stop; // clock\_t is a valid type

double duration;

The additional statements are used to time the function you wrote computing the mode(s).

start=clock(); // Start the benchmark

// call the function that is to be timed

mode(data, size, numModes, modeNums, maxFrequency);

stop=clock(); // stop the clock

time1 = double(stop-start)/CLK\_TCK; // compute/record the time

cout<<"time to execute the mode "<<time1<<endl; // output the time to compute the mode(s)

**Files to submit no later than March 27, 2016**

Create a Word document that will be submitted at the same time you submit the cpp file. This file must contain your name, your CWID, your class standing (undergraduate or graduate), and your email address. In this document briefly describe the algorithm you used to solve the programming challenge. Also list the time it took your program to execute the TestData file. Name your Word file with your name as follows – “LastNameFirstName” such as CreiderDaniel. Use the same naming convention for your cpp file and be sure to put your name in comments in the cpp file. Do not forget that there is a deadline for submission of your code.

When you submit, you are certifying that you wrote the code that was submitted and you did not submit code written by someone else and put your name on it. If you have any questions about the code to read a binary file or to time the execution speed of your program, send me an email or see me in person in my office. I will help resolve the problem with your code for these issues. All questions about this programming challenge should be directed to me.

Dr. Dan Creider - [Daniel.Creider@tamuc.edu](mailto:Daniel.Creider@tamuc.edu) - Office phone - 903-886-5407 - Jour 216