CSCE 2004 – Programming Project 5
Midpoint Due Date – Apr 5, 2017 at 11:59pm
Final Due Date – Apr 12, 2017 at 11:59pm

1. Problem Statement:

The primary goal of this programming assignment is to give students experience with two-dimensional arrays. Your task will be to implement some of the basic number crunching features that made early spreadsheet programs VisiCalc and Lotus 1-2-3 the first "killer apps" for Apple and IBM back in the 1980s.

As you know, spreadsheets allow you to store data values in a two-dimensional grid of cells. In Excel, the locations of cells are given by integer row indices (1,2,3,…) and character column indices (A,B,C,…). Using this notation, you can create formulas that combine the data from a range of rows and columns. For example, if you have 10 values in column A from row 1 to row 10, you can add these up using the formula "SUM(A1:A10)". If you experiment with Excel, you will discover that they support hundreds of numerical formulas, which explains why the implementation of Excel is estimated to be hundreds of thousands of lines of code.

Your task in this programming assignment is to make a very simple menu-based spreadsheet program called "MiniCalc". To keep things simple, your spreadsheet should store all data in a fixed size float array of 26 rows and 26 columns. Your program should index these locations using Excel's convention with row indices [1..26] and column indices [A..Z]. Your program should support the following spreadsheet operations:

- Store <u>specified</u> data value in cells from (col1, row1) to (col2, row2).
- Store <u>random</u> data values in cells from (col1, row1) to (col2, row2).
- Calculate <u>minimum</u> of data values from (col1, row1) to (col2, row2).
- Calculate <u>maximum</u> of data values from (col1, row1) to (col2, row2).
- Calculate <u>sum</u> of data values from (col1, row1) to (col2, row2).
- Calculate <u>product</u> of data values from (col1, row1) to (col2, row2).
- Calculate <u>average</u> of data values from (col1, row1) to (col2, row2).
- Calculate <u>standard deviation</u> of data values from (col1, row1) to (col2, row2).
- <u>Print</u> data values from (col1, row1) to (col2, row2).

In each of these operations, you must prompt the user for the range of array locations to process (col1, row1) and (col2, row2). You must check that the user enters row values between [1..26] and column values between [A..Z], and then convert these to actual array indices between [0..25]. You should also check that col1 <= col2 and row1 <= row2. Once the input parameters are verified, your program should calculate and print the result.

For the two operations to store data, you should prompt the user for the specified data value, or the [low..high] range for the random data values, and use this information to initialize the specified locations in the data array.

For this assignment, students are allowed to use any combination of C++ features we have discussed in class or lab. You are not required or allowed to look ahead to more advanced C++ features like files or classes to perform this task.

2. Design:

For this assignment, you have two big design decisions. First, you need to decide what the user interface for the program will look like. For example, what menu will you use for the operations above? How will you read and verify the input parameters for each operation? Finally, what to do if the user enters incorrect information?

The second design decision is how to implement each of the operations above. Do you want to use small functions that take an input array and other parameters? Do you want to put the code inside the processing menu? What formula is needed to calculate random integer values in the range [low..high]? In order to calculate the median of the specified values, you will need to copy the selected data into a temporary array and then sort it. What sorting method will you use to do this? Finally, what formula is needed to calculate the standard deviation of N data values?

3. Implementation:

You are starting this programming project with a blank piece of paper. Depending on your design choices above, you could start by implementing the user interface portion first, printing menus, reading and verifying parameters, and get this all working before you implement the operations above.

Another option might be to write and debug some small functions to perform the operations above, and "hard code" calls to these functions in the main program to test them. Once they are working, you can add the user interface to get user input and call one of these functions to do the work.

In either approach, it is very important to make these changes incrementally one feature at a time, writing comments, adding code, compiling, and debugging. This way, you always have a program that "does something" even if it is not complete.

4. Testing:

Test your program to check that it operates correctly for all of the requirements listed above. Also check for the error handling capabilities of the code. Try your program with several input values, and save your testing output in text files for inclusion in your project report.

5. Documentation:

When you have completed your C++ program, write a short report using the "Programming Project Report Template" describing what the objectives were, what you did, and the status of the program. Does it work properly for all test cases? Are there any known problems? Save this project report in a separate document to be submitted electronically.

6. Project Submission:

In this class, we will be using electronic project submission to make sure that all students hand their programming projects and labs on time, and to perform automatic plagiarism analysis of all programs that are submitted.

When you have completed the tasks above go to Blackboard to upload your documentation (a single docx or pdf file), and your C++ program (a single cpp or txt file). Do NOT upload an executable version of your program.

The dates on your electronic submission will be used to verify that you met the due date above. All late projects will receive reduced credit:

> 10% off if less than 1 day late,
> 20% off if less than 2 days late,
> 30% off if less than 3 days late,
> no credit if more than 3 days late.

You will receive partial credit for all programs that compile even if they do not meet all program requirements, so handing projects in on time is highly recommended.

7. Academic Honesty Statement:

Students are expected to submit their own work on all programming projects, unless group projects have been explicitly assigned. Students are NOT allowed to distribute code to each other, or copy code from another individual or website. Students ARE allowed to use any materials on the class website, or in the textbook, or ask the instructor and/or GTAs for assistance.

This course will be using highly effective program comparison software to calculate the similarity of all programs to each other, and to homework assignments from previous semesters. Please do not be tempted to plagiarize from another student.

Violations of the policies above will be reported to the Provost's office and may result in a ZERO on the programming project, an F in the class, or suspension from the university, depending on the severity of the violation and any history of prior violations.