CSCI-14 Lab 9 – due 4/17/18

Please work in groups of two on this. There are three programs.

Program 1 (easy):

Write a program that asks the user for exactly 50 integers, and stores them in an array. Then print the largest, smallest, sum and (floating-point) average of the entries. After this, print the numbers in the entered order, then in the reverse order (print in 5 rows of 10 numbers each, use setw() to align), with reasonable text before each group to describe what you're printing. Then (only **after** finding largest and smallest, sum and average, and printing the array twice) sort the array (using a simple sorting algorithm like ripple, bubble or selection sort from chapter 8 in the book), and print the entries in ascending sorted order. You may assume the entries will each be less than 1000 and greater than -1000. Test with at least two different series.

Program 2 (slightly trickier):

Write a program that asks the user no more than for 50 numbers (integers), stopping at either the 50th integer or the first non-positive entry (<= 0, the sentinel), storing them in an array. Again, print the largest, smallest, sum and (floating-point) average of the entries, if these exist, otherwise a message that there are no statistics. After this (again, only if you have data), print the numbers in the entered order, then in the reverse order (print in up to 5 rows of at most 10 numbers each, use setw() to align) with reasonable text before each group to describe what you are printing. When printing, the last row might not have 10 elements. Then, if you have data, (again, only **after** finding largest and smallest, sum and average, and printing the array twice) sort the array (using a simple sorting algorithm like bubble sort or selection sort from chapter 8 in the book), and print the entries in ascending sorted order. You may assume the entries before the stop sentinel will each be less than 1000. Test with at least several different series, including series with zero entries, with less than 50 entries and with 50 entries. Do not count, store in the array, or otherwise process the sentinel at all.

Program 3 (slightly trickier again – think carefully about your main loop design):

Write another similar program, but stop taking entries only at the first value less than or equal to 0 (the sentinel). The user may enter more than, exactly, or fewer than 50 entries before the sentinel. Ignore any entries greater than 0 after the 50th entry, but continue taking entries until the user enters a value less than or equal to 0. Do not count the extra entries or try to store them in your array. Process only those values before the sentinel or in the first 50 values if you don't see the sentinel before the 50th entry. When printing, the last row might not have 10 elements. Otherwise, you need to do the same things with the data set in the same sequence as before. The entries you process must all be positive (greater than 0) and you may assume they will be less than 1000. Test (at least) with a series of zero entries, less than 50 entries, exactly 50 entries, and more than 50 entries. This is slightly trickier than #2, but not much – think clearly about how to design the loop (singular – one loop is all you need) for your input!

Turn in the three programs with their tests.

Some comments: you may use input and output redirection on the command line (program < datafile > outputfile) to get the data into and out of the programs. Some (but not all) installations of Code::Blocks will not let you run the executable from the command line (the configuration may not be set up to find a library needed to run the executable), so you might want to use Quincy for this if you want to use the command line for redirecting I/O. There is a fix for this problem posted on my CSCI-20 Web site for this problem. You may also prompt for the file names and do direct file I/O in the programs. In either case, name the data and output files in a way that shows their relationship clearly. You may also just accept values from the keyboard and capture the CMD shell text to turn in, but I think this is the hard way to do this...

Think the problems through (design them!) before you code these!