CISC 1110 Spring, 2017 Assignment 5 Due April 10, 2017

Write a complete C++ program to do the following: The main program will call a series of functions to process a set of data. One function will read data into two arrays. A second function will print the values stored in an array. A third function will find the smallest of the values stored in an array. A fourth function will construct a new array from two existing arrays. A fifth function will compare the values in the two arrays. **Use files for both input and output.**

1. The main program will send n and the two arrays as parameters to the function **readdata**; **readdata** will read in n and then read n items into two arrays (which the main program will call score1 and score2). All data values must be read from a file, and all values—including n--must be read by the function.

2. The main program will call a function **printarray** (twice, once for array score1 and once for array score2) to print the values stored in the two arrays. Each time before the main program calls the printing function, it should print a heading saying what is being printed (for example, it could say "score1 array"). All output, including headings, should go to a file.

3. For each array, the main program will call a function **smallest** which will find the smallest of the n values stored in an array. The main program will call the function once to find the smallest of the n values stored in the score1 array, and once to find the smallest of the n values stored in the score2 array. The main program will print the returned value, together with a relevant message.

4. The main program will call a function **construct**. The function will construct a new array (which the main program will call sumscore) from the two arrays score1 and score2. Then **construct** will call **printarray** to print the sumscore array. Either main or the function **construct** can print the heading for the sumscore array.

5. The main program will call the function **smallest** again to find and return the smallest of the n values stored in the sumscore array. The main program will print this value.

6. The main program will call a function **whatshigher.** This function will be used to compare the contents of the score1 and score2 arrays. The function will receive six parameters: two arrays (score1 and score2**)**, an integer n (giving the size of the two arrays), and three counters, which are reference parameters. The function will compare the two arrays, element by element, to calculate in which array the value is higher. In addition, the function will change its three counter parameters; after the function returns to main, main will print the counters, together with clear messages.

Here are the details of the functions.

A. The function **readdata** will read all the data into two arrays. The function will receive three parameters: an integer which it calls n, and two arrays which it calls nums1 and nums2. At the beginning, none of the parameters will have values. The value of n will be used to control the action of the function, as described below.

The function will read in n and then read in n sets of data. Each set of data should contain two integers; the first will be read into nums1, and the second will be read into nums2. (The values stored in these parameters will remain when the function returns to the main program, and they can be used throughout the program.) For example, suppose the data set is the following:

```
3
18  13
21  42
54  66
```

The **readdata** function will read the value 3 into the variable n. Then it will read the values 18, 21, and 54 into the array nums1 and the values 13, 42, 66 into the array nums2. Note that you cannot read down the column of values–the two values you read will come from the same line, and you will place the values into the two arrays.

B. The function **printarray** will receive two parameters, an integer which it calls lim, and an array which it calls vals. It will print the lim values stored in the array vals, all on one line, with spaces between the values.

C. The function **smallest** will receive two parameters, an integer which it calls num and an array which it calls arr. It will find and return the smallest of the first num elements of the arr array.

D. The function **construct** will receive four parameters: an integer which it calls k and three arrays which it calls old1, old2, and summ. The function will construct the first k elements of the summ array from the first k elements of the old1 and old2 arrays, as follows. Each element of summ will be equal to the sum of the corresponding elements of old1 and old2. For example, summ[0] will be old1[0] + old2[0], summ[1] will be old1[1] + old2[1] and so on. (You are using parallel arrays.)

E. The function **whatshigher** will receive six parameters: two arrays (which it calls score1 and score2**)**, an integer which it calls n (giving the size of the two arrays), and three counters. The function will compare the two arrays, element by element, to calculate in which array the value is higher. For each array element, the function will print which array has the higher value (or if the two values are equal). The function should print each of the values, the array it is from, and the relationship. For example, if score1 contains 75 76 77 and score2 contains 67 86 77, the function should print something like the following messages:

```
in position 0 the higher value is in array score1: 75 is higher than 67
in position 1 the higher value is in array score2: 86 is higher than 76
in position 2 the value is the same in both arrays: 77
```

In addition, the function changes its three counter parameters: the three counters tell how many times the value from score1 is larger, how many times the value from score2 is larger, and how many times the values in the two arrays are equal. The counters should get all their values, **including their initial values**, in the functions.

DATA: Please read the data from a file which is to be prepared in advance (be sure to print the file and submit it). Use a parameter value of 10-15 (but write a program which will work for any size arrays up to 50 or so).

Make sure that the smallest element does not occur in the first or the last position of an array. Make sure that the smallest element does not occur at the same position in each array. For example, the smallest might be at position 5 in array score1, it might be at position 3 in array score2, and it might be at position 6 in array sum. Make sure that the two arrays have the same value in at least one position. Try to make sure that the values make the counters printed at the end have 3 different values (below, my final counts are all different).

OPTIONAL

Modify the program so that, after processing a complete set of data, it will go back and repeat the entire process for a new set of data, and so on, until the end of the data file. If you do this optional, make sure that you run it with a set of data values that show that it works.

SAMPLE INPUT AND OUTPUT:

A sample data file (yours will be larger):

```
6
95     94
98     98
42       45
43       18
64       54
83       79
```

Sample output for this data file (your order and your words might be slightly different):

```
There are 5 values in each array.
The score1 array:
95      98      42      43    64      83
The score2 array:
94      98      45      18    54    79

The smallest value in score1 is 42
The smallest value in score2 is 18

The sumscore array: 189    196    97      61    118      162

The smallest value in sumscore is 61

in position 0, array score1 is larger: 95 is greater than 94
in position 1, the two arrays have the same value: 98
in position 2, array score2 is larger: 45 is greater than 42
in position 3, array score1 is larger: 43 is greater than 18
in position 4, array score1 is larger: 64 is greater than 54
in position 5, array score2 is larger: 83 is greater than 79

array score1 was larger 3 time(s)
array score2 was larger 2 time(s)
the two arrays were equal 1 time(s)
```