**CSIS 112: Lab 3—Sort Scores**

**Instructions:**

This assignment utilizes a class in an application that might be useful in the “real world.” It requires the sorting of, and computation with, data involving test scores.

Create a class (Scores) that stores test scores (integers) in an array (assume a maximum of 100 scores). The class should have a constructor that allows the client to specify the initial value of the scores (the same initial value applies to all of them) and another default constructor that initializes them to 0. The class should have functions as follows:

1. A member function that adds a score to the array. The client must *not* pass the array index to the object. This member function *returns* an error indication if the value is invalid – i.e. negative test scores are not permitted.

2. A member function to sort the array in ascending order.

You may find this site to be helpful in writing your sort function:

<http://www.cplusplus.com/articles/NhA0RXSz/>

3. A member function to compute the average of the scores in the array, and

4. A member function to display the scores.

Note that you will need to be careful about sorting the array and computing the average of the scores. Because your constructor will initialize all members of the array to be a value, you do not want to include these values in your average calculation nor in your sort function.

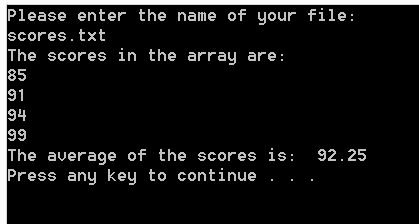
Write a program (client) that uses the class by creating a Scores object and prompting the user for a file name. Appropriate error checking is required to ensure that the file exists and can be opened successfully.

The client should read the file contents and store them in the object. The file will be formatted so that the first line contains the number of scores, and each successive line contains a score. A typical input file might contain:

* 4
* 90
* 85
* 99
* 94

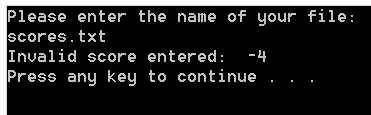
The client should read and store the contents of the file, sort and display the scores, and compute and display the average score. All output should be labeled appropriately, and validity checking should be done on input and storage of the data. If an error is encountered in the input, the program should output an error message indicating which item was invalid, and the entire program should then terminate.

The executing program should look something like this:

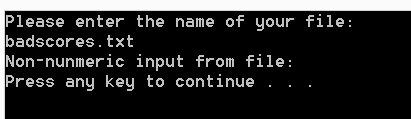


In the screen above, notice that the average is displayed with decimal points. Whenever you are calculating an average, never truncate the decimal portion until you are told to do so. In this lab, you are not to truncate the average.

If a negative score is read in, the program might look something like this:



If a non-numeric score is read in, the program might look something like this:



In the screen above, notice that the program pauses long enough to read the error message that is displayed.

Use good coding style (modular, separate .h and .cpp class files, no globals, meaningful comments, etc.) throughout your program.

**Deliverables**:

* Complete the programming assignment described above and submit your completed assignment in accordance with the lab submission policies.