CSIS 112: Lab 4—Pointers

*[Adapted from Dr. Terry Metzgar and Starting Out with C++, Tony Gaddis, 4th ed, Scott/Jones, Inc, pg 560.]*

**Instructions**:

Create a class called TestScores that stores test scores (integers) in a vector.

The class should have a function that adds a score to the vector (call it addScore), another that sorts the vector in ascending order (call it sortVector), a third member function that computes the average of the scores in the vector (call it computeAverage), and a fourth member function that prints the scores to the screen.

This is a lab that reinforces pointers. To receive credit for your lab, in any function that requires arguments, those arguments should be declared as pointers in your function headers. At least one of your functions should require an argument.

Write a program that uses the class to perform the following features: [Note: The program should have a menu, and these are the items your menu should include.]

1. Read scores from a file.

[Note that when the user selects this option, the program should prompt the user to enter a file name to open. It should then open the file and read it into the vector by calling the function that adds a score to the vector. Hint: Make sure that the file exists before trying to read from it.]

1. Add a score from the keyboard.

[Here, just prompt the user for a score and call a member function to add the score to the vector.]

1. Sort the scores. [Write your own algorithm for this. Do not use any built-in functions to sort the vector.]
2. Compute the average score. [Again, write your own algorithm for this.]
3. Print the scores to the screen.
4. Exit the program.

If the user chooses to read the scores from a file, you may assume that the file will be formatted such that every line contains a score. For example, an input file might contain:

95

77

88

99

Read scores into your program until the end of file is reached. A file may contain 1 score or a million scores. The program should be robust enough to handle as many scores as are in the file.

Use appropriate error checking. Integers are expected as scores; therefore, make sure that the user enters appropriate data types. If the user enters a number with a decimal point, be sure to produce an appropriate error message and allow the user to re-enter the score. Negative scores should not be allowed.

**Deliverables**:

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