ECE 3220 Lab 9 Report

John Walter

November 10, 2016

Objective

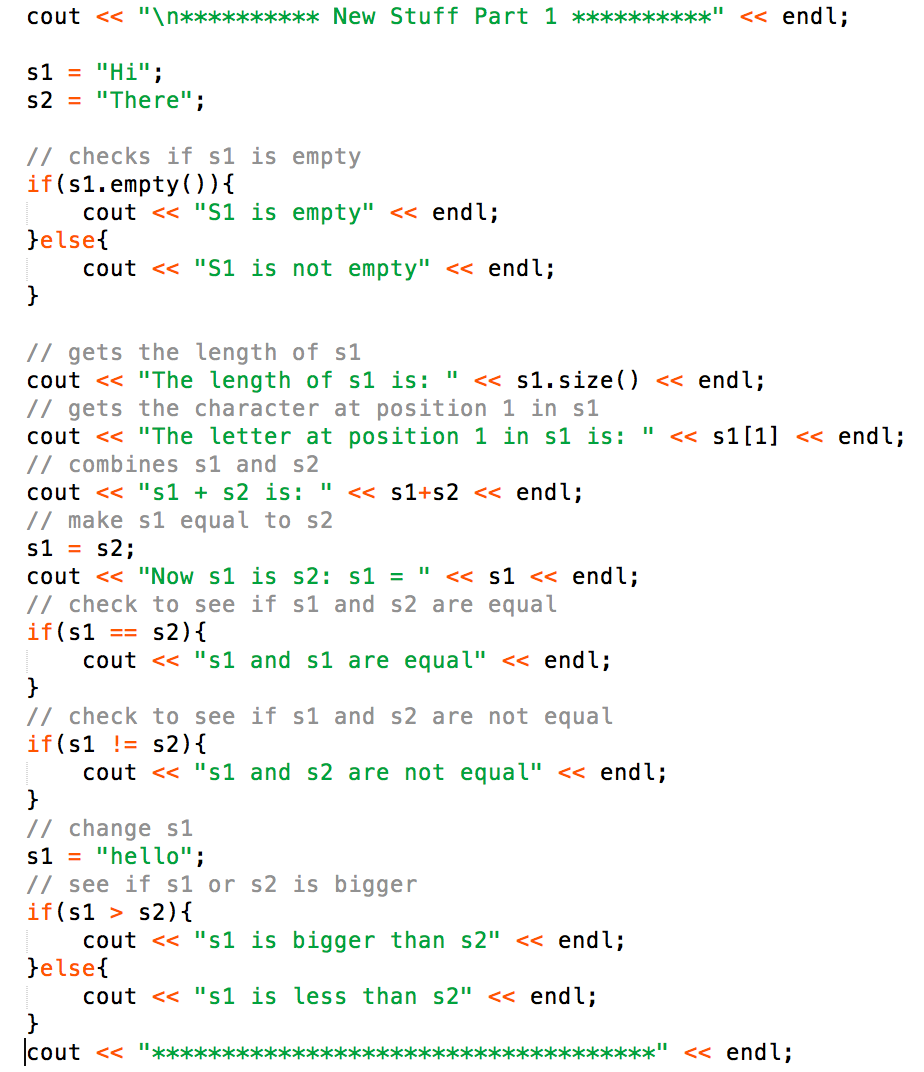
The objective of this lab involved the review of several concepts in the C++ programming language. The first of which was to become more familiar with how strings can be used and manipulated to perform certain tasks more simply. The next topic covered was to learn about how vectors can be declared and used within a program, and also specific functions related to vectors that are useful during implementation. The final topic reviewed in this lab was the concept of operators in C++. This involved the process of overloading an already standard operator with predefined specifications to perform more specific tasks.

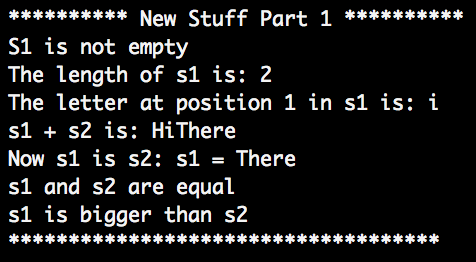
Results

This first part of this lab was to review some already written code and add certain operations defined in the book.

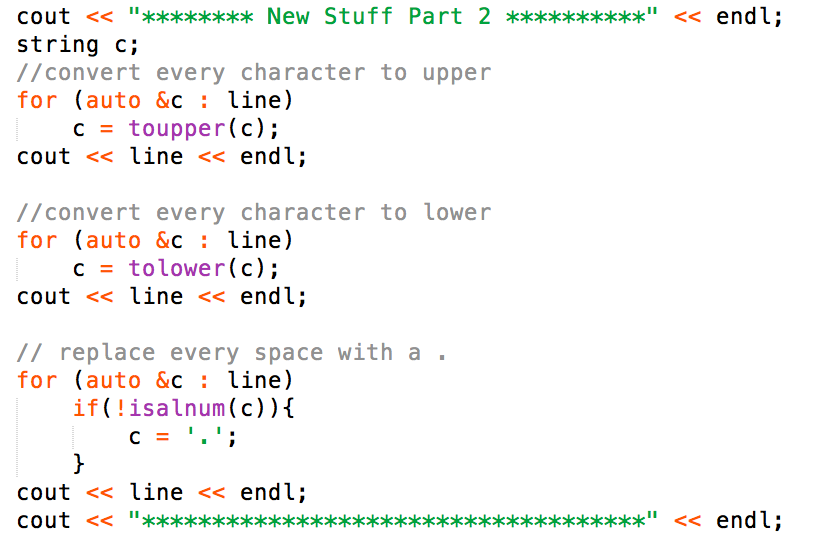
For Lab9\_strings.cpp:

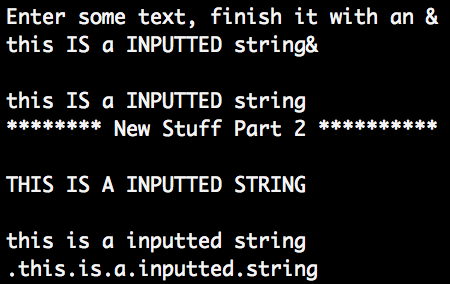
The first part of this section was to review some string operations given in a table in the book. The results are as follows:





These results were straightforward and as expected. The second part of this section was to use a “Range for” and a few operations to (1) convert the input string to lowercase, (2) convert the input string to uppercase, and (3) replace each space with a “.”. The results are as follows.

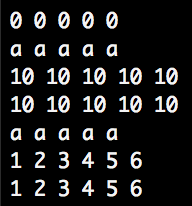
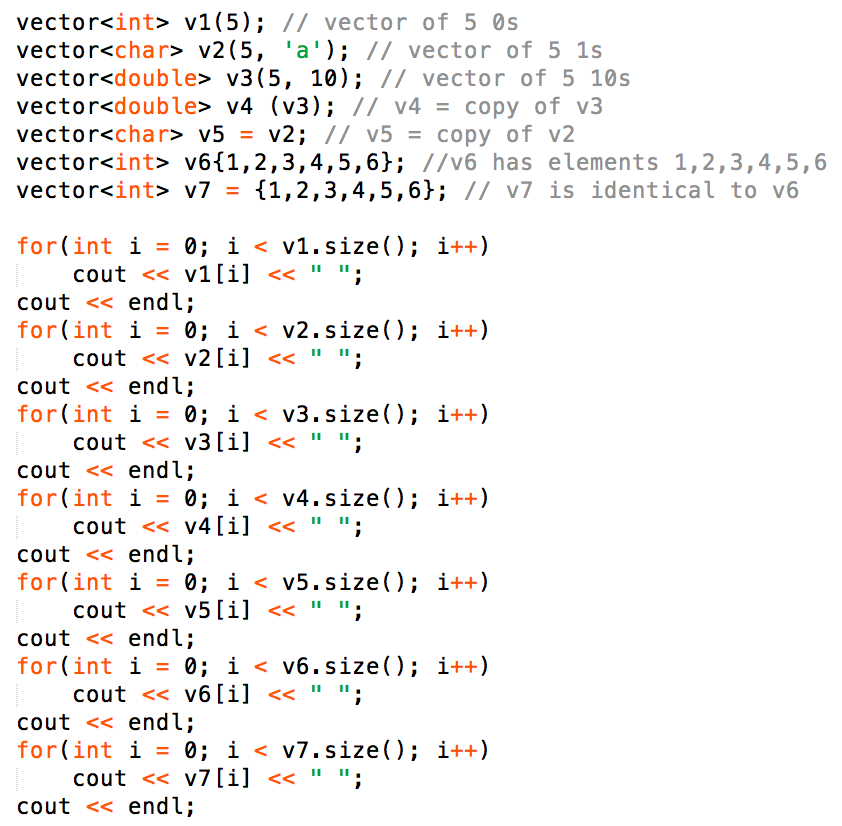




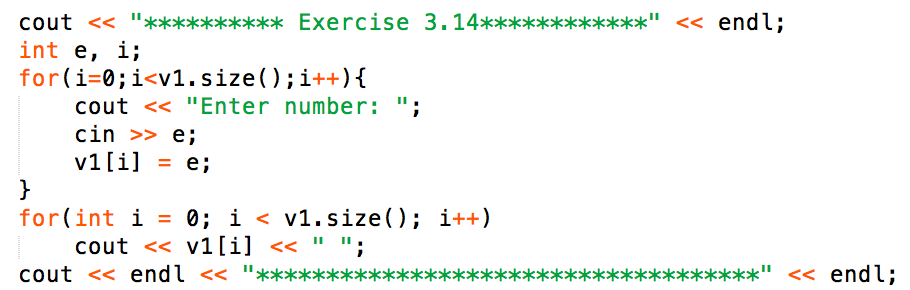
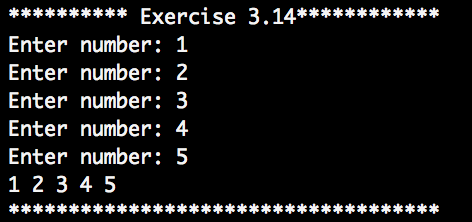
Again, the results follow exactly what was expected. The available string operations made these tasks much simpler to implement.

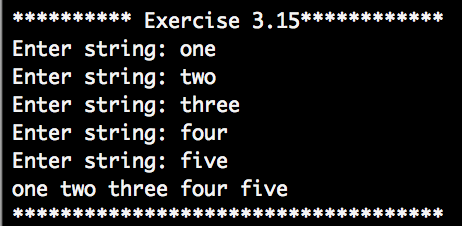
For Lab9\_vectors.cpp:

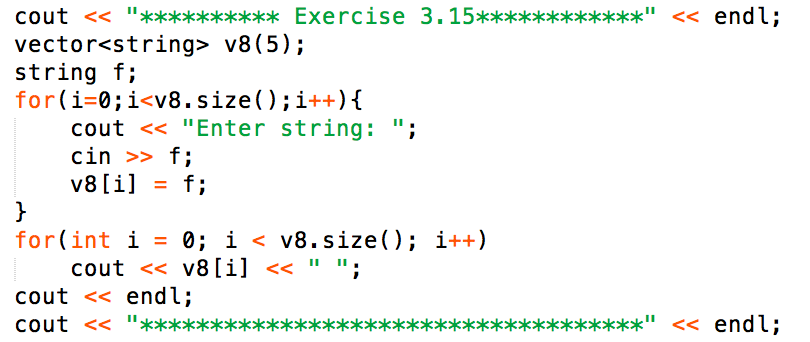
The next part of this lab was to review how vectors can be declared and some of the operations specific to them. This first part was to define some vectors using some operations given in the table in the book.



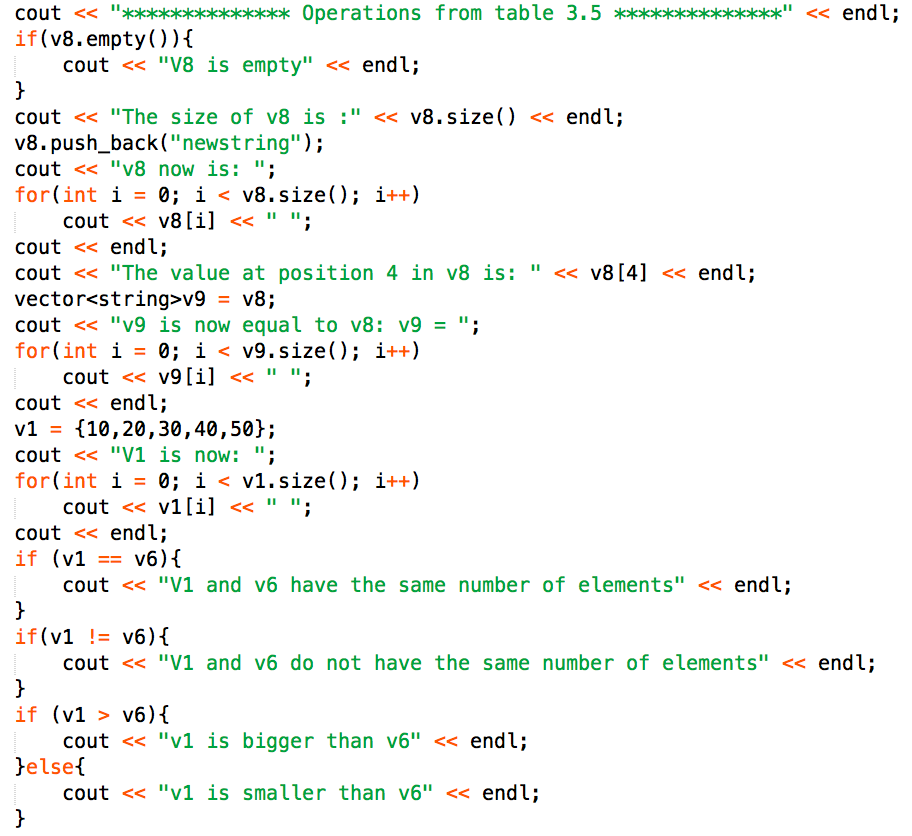
The second part of this section was to complete a couple of exercise from the book. The first one to read in some integers and store them in a vector.

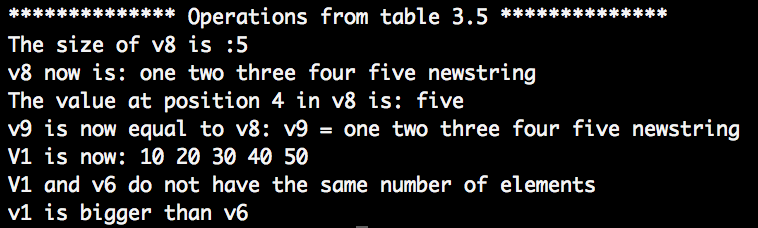


The second one was similar except instead of scanning in integers, strings were scanned in and stored in a vector of type string.



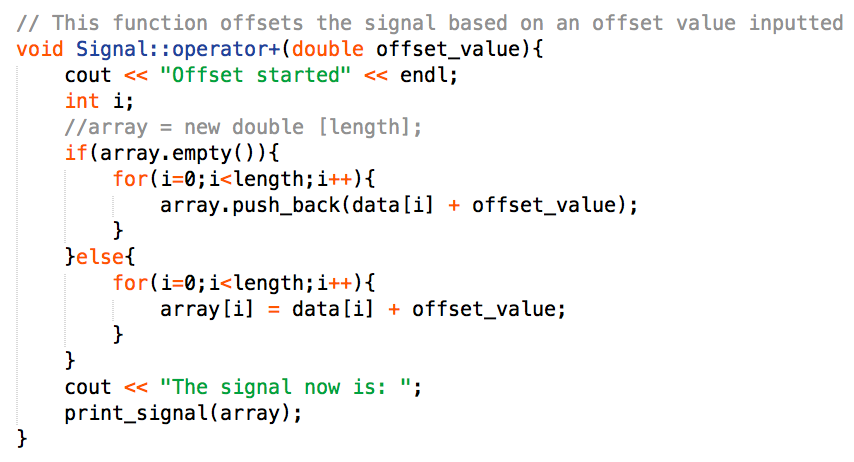
The last part of this section was to review some operations given in the book.





The final part of part one of the lab was to analyze some code dealing with operators in C++ and observe the output. My answers to the questions are included in the comments of the source code.

Part two of the lab involved transforming the program from lab 7 into using vectors and overloaded operators. To do this, the offset method was transformed into the operator + that would take in the offset value and then preform the offset.

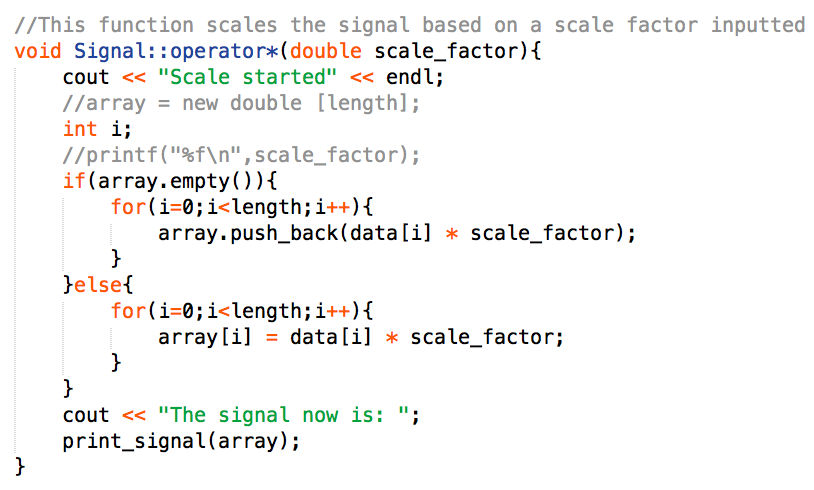


This can then be implemented by:

../../../../../Desktop/Screen%20Shot%202016-11-09%20at%2011.3

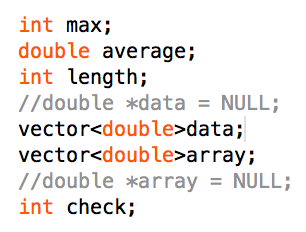
Because the array is initially empty, the function push\_back() needed to be implemented because the array had no length. However, once a transformation had been done, the array is not empty and therefore array[i] can be used.

Similarly, the scale method was transformed into the operator \* that would take in the scale factor and then preform the scale.

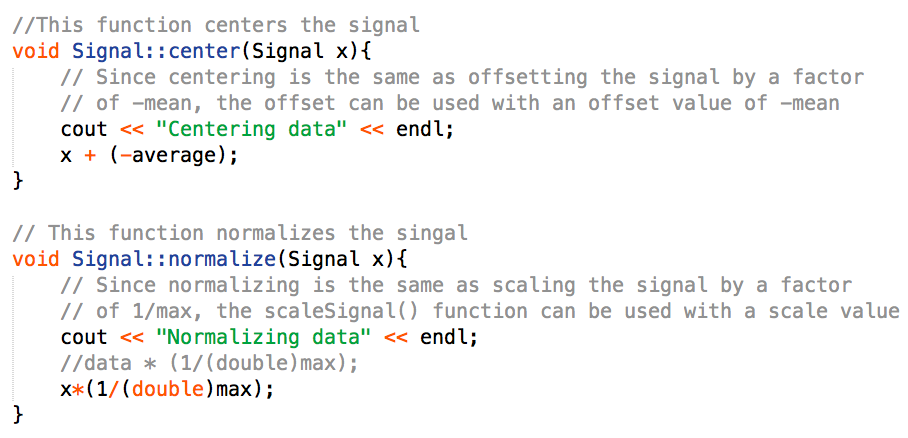


This can then be implemented by:

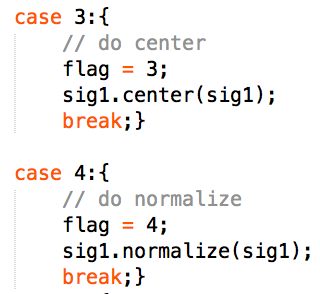
../../../../../Desktop/Screen%20Shot%202016-11-09%20at%2011.4

The declaration of the pointers were then converted into vectors by:

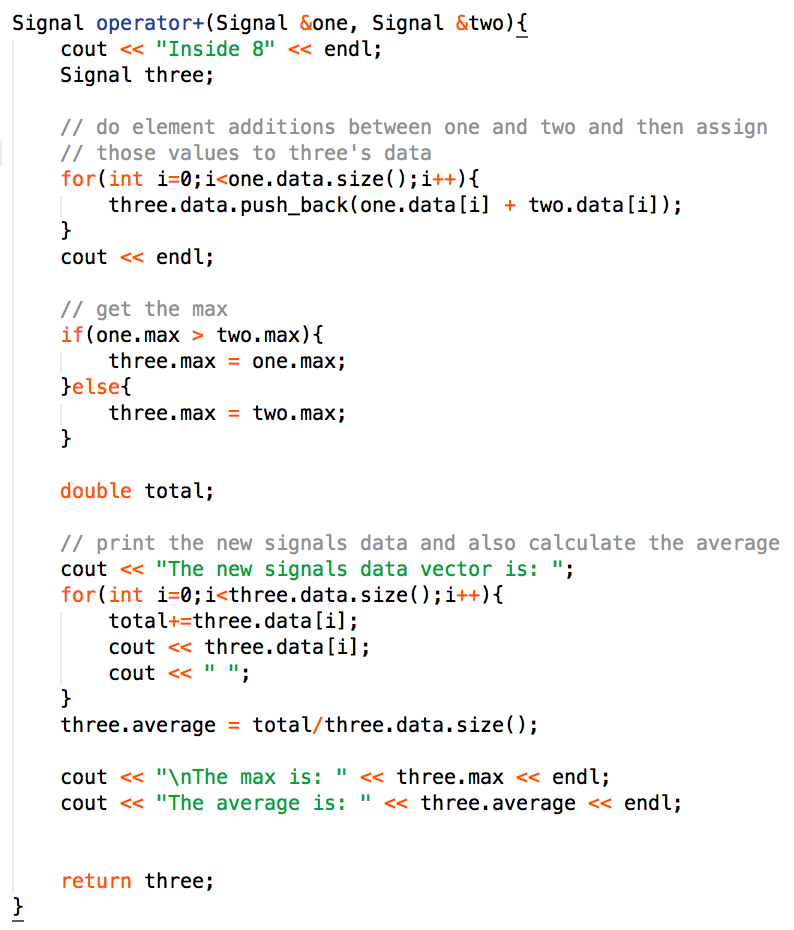
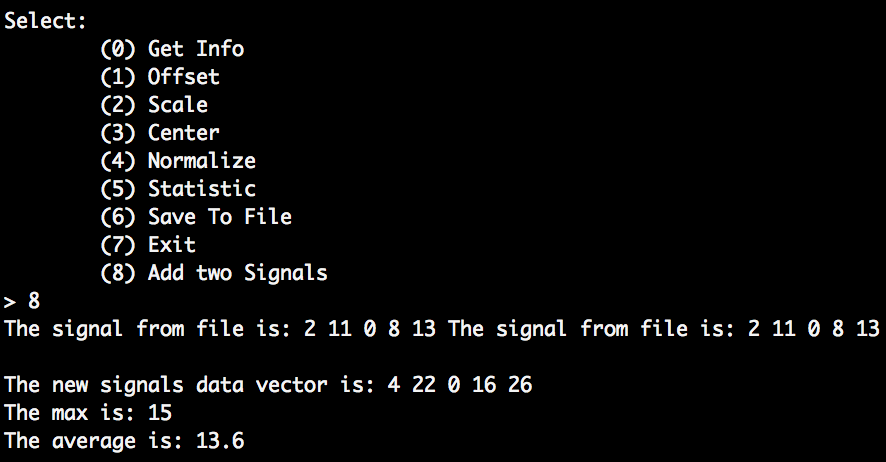
These new operators could then be used within the normalize method and the center method by passing the Signal object to each method to perform the operations.



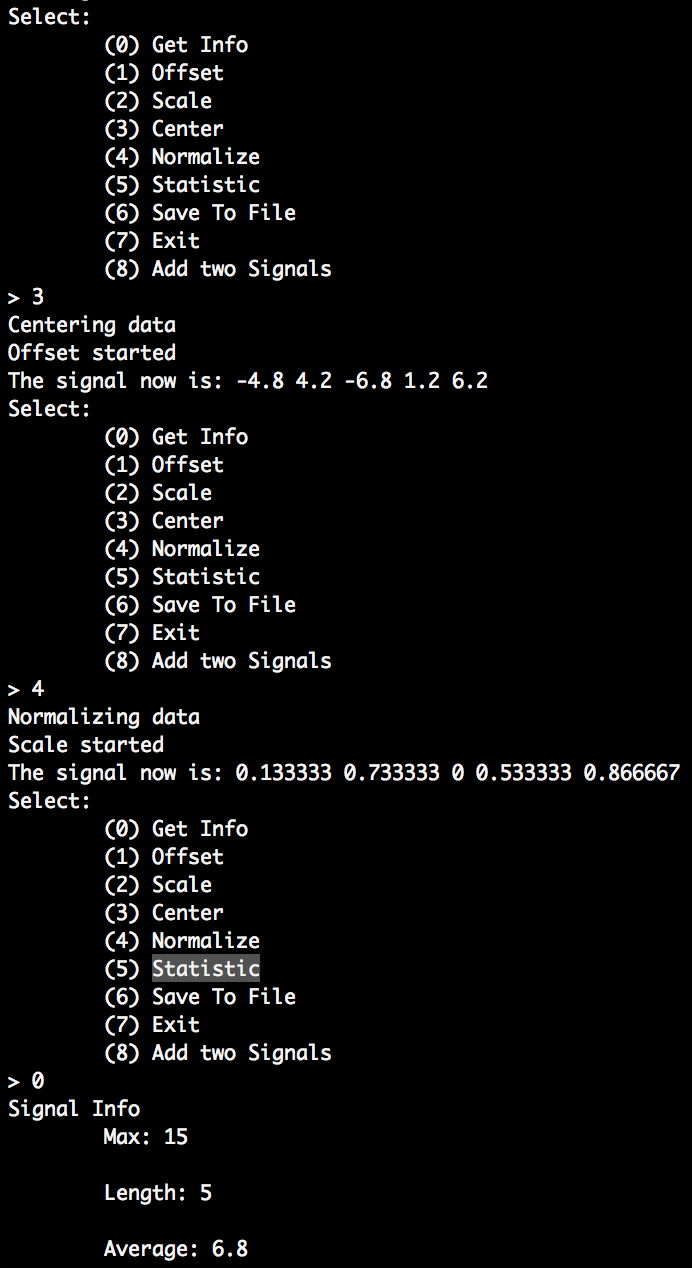
These are then called by:

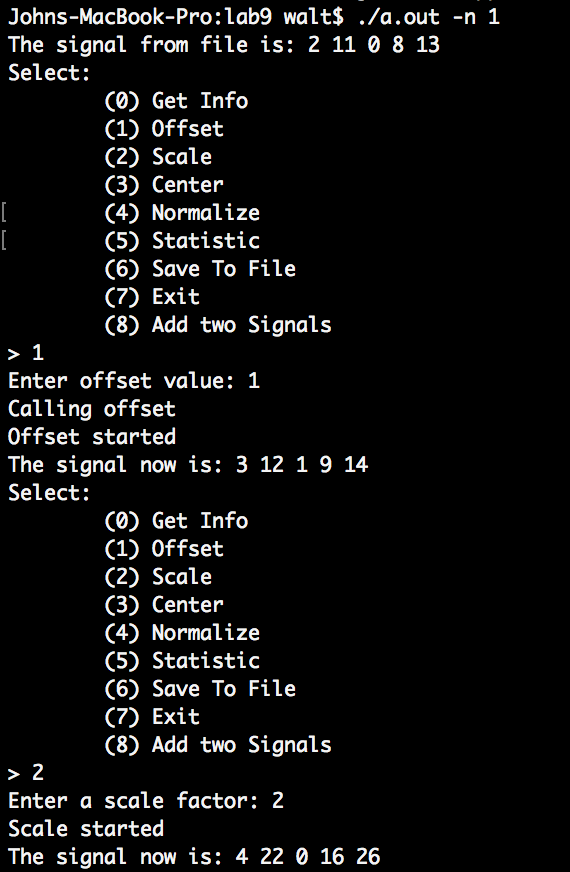


The last part was to create a non-member addition operator that takes in two Signal objects and returns a Signal object with a specific combination of the two Signals. This can be found within the menu (8).



The two “The signal from file is:…” are the initialization of each Signal object. It should be noted that objects “one” and “two” both opened the same file, so the new objects data will contain double that of the original signal.

Program output:



It should be noted that each transformation is performed on the original signal each time and not the transformed signal.

Discussion

Everything in this lab was fairly straightforward, however one issue did occur. The issue happened when trying to use the defined operators + or \* within the normalize and center functions. It was not obvious how to access the object to do the “addition with” (the first operand). The only feasible way encountered was to directly pass the object as a parameter to the method in order to have a first operand of type Signal. Once that was completed, the program ran as expected.