Homework 5
Due Date: April 20
9 Points Total

We've seen the big-O notations for comparing the different sorting methods, but how do they really perform? Often when you devise some new algorithm (and publish it), you include the run-time estimate (big-O) as well as real statistics, e.g. "execution of the World's Greatest Sorting Algorithm was 0.35s on 10 bazillion elements", or something like that.

So let's see how these sorting methods really perform on different types of data. I've included four files:

- Random.txt,
- Reversed.txt,
- NearlySorted.txt,
- FewUnique.txt.

Each file contains 10,000 integers (between the values of 1-10,000) which are randomly distributed, in descending order, nearly sorted in ascending order, and very few unique values, respectively.

Your task is to implement the following algorithms:

- Bubble sort
- Insertion sort
- Quick sort
- Shell sort
- Merge Sort

For each of these sorting strategies you will execute the sort and determine the number of comparisons and the number of exchanges required to sort each of the file types. (Hint: your textbook is a great resource.)

More specifically, what do you need to do:

1. Create a function to read in the integer data from the supplied files.
2. Implement the five sorting strategies.
3. Verify using small test sets (create them yourself) that they are in fact sorting the data correctly. (Check out http://www.random.org/integers/ to help generate random numbers.)
4. Execute the code and identify the number of comparisons and the number of exchanges.

Your submission will include 2 things:

- Your source code. Please put all of the necessary functions as well as the main function in one *.cpp file. There is no need to create a class or header files. ***This will be submitted via GitHub.***
- The table which includes your trials in the table. (HW5-table.docx). ***This will be submitted via Sakai.***

Grading:
5pts (1pt each) Functionality of the five sorting techniques.
1pt Quality commenting
3pts Correct and complete table