CS 2322
Lab 6

Sorting Algorithms I

Implement the following projects and collect any results in a Word (or other word-processing) doc to be turned in to D2L by the due-date. Copy any questions, in **bold**, into your document so that I have context for your answers. Title and label axes of all graphs, if graphs are required.

Part I:

- BubbleSort
- SelectionSort
- InsertionSort
- ShellSort

Implement the sorting algorithms above, and generate average timings on random Lists of 10 equidistant sizes of ***fraction*** objects ***as well as*** same-size lists of unique ***integers***.
(**Hint:** should you want to print out list-contents to ensure your sort-algorithms are actually sorting, you may want to use your C++ Fraction class's `getVal()` method to return the value of a floating point number rounded to about three decimal places, rather than the numerator/denominator format)

- Suggestion: Create a large vector (roughly 10x the base-length of your ten list-sizes) of randomly-ordered integers. Also, build a second similarly-large vector of randomly-generated Fraction objects (as in your Lab 1). Pass a *sublist* or *copy* of your randomly-ordered vector to each sorting-function

- Why do I suggest passing a ***`subset`*** of your large vector to the sorting function, rather than simply passing a randomly-generated vector of the appropriate size? Because you don't want to sort the numbers in the original, randomly-ordered list. This would give subsequent algorithms to which you pass the sorted list a timing-advantage!
- Collect timings of ten evenly-spaced list-sizes (e.g. `10000 - 100001`), *using the same set of values for each sort-algorithm*. Compare same-size fraction-sort times to integer-sort times. Represent the results of your timings in one or more Excel charts, to best illustrate any differences in each algorithm-class.
  Is sorting one type of data consistently faster/slower than another? Why do you think that might be?
- Explain the results shown by your charts.

Hand in a hard copy of your report to D2L.