CIS2168 006 Assignment 7
Sorting

1. Objectives

This assignment will help you to:

- Understand how sorting algorithms work
- Explain the performance differences of different sorting algorithms

2. Description

You will write a number of programs.

2.1 Write a method mergeSort(), which merge-sorts an array of integers of type `int` and keeps track of **msMoves**, the number of moves required to sort an array**.** Write a class MergeSortTest, which asks a user for an array size, fills the array with random integers that are within the range of [0, 5000), calls mergeSort() to sort the array, and prints the value of **msMoves**.

2.2 Write a method quicksort(), which quick-sorts an array of integers of type `int` and keeps track of **qsCount**, the number of comparisons of ***array*** entries required to sort an array. Write a class named QuickSortTest, which asks a user for an array size, fills the array with random integers that are within the range of [0, 5000), calls quickSort() to sort the array, and prints the value of **qsCount**.

2.3 Write a method insertionSort(), which insertion-sorts and keeps track of **isCount**, the number of comparisons of ***array*** entries required to sort an array. Write a class named InsertionSortTest, which asks a user for an array size, fills the array with random integers that are within the range of [0, 5000), calls insertionSort() to sort the array, and prints the value of **isCount**.

2.4 Write a program named SortComparisionTable that prints a table for each of the above sorts. The table must have three entries for each of these array sizes: 2048, 4096, 8192, 16,384.

- For example, for mergeSort,
  - The tree entries in its table for a given size, say 2048, are determined by running MergeSortTest 10 times, resulting in 10 values for **msMoves**.
  - The three entries are then the minimum of these 10 values, the average of these 10 values, and the maximum of these 10 values.
  - So if the values were 2, 8, 6, 3, 12, 6, 7, 2, 4, 8, then the three entries would be 2, 5.8 = 58/10.0, and 12.
- For quick sort and insertion sort, these three entries would involve the 10 values for **qsCount** and **isCount** respectively.
- Your program must use a variable of k runs, where users enter k, not just constant 10 runs.

3. Implementation Requirement

- Your sorting methods must sorts a list of int values, not Integer objects. So for merge sort, it is
  - public static void mergeSort(int[] array)
    i. Add recursive private methods if necessary.
- The classes must use Scanner to read-in the array sizes.

4. Major Steps

- Understand the related classes I gave you in Lec#11.
- Revise InsertionSort and TestSort to be InsertionSortTest.
- Revise MergeSort and TestSort to be MergeSortTest.
- Revise QuickSort and TestSort to be QuickSortTest.
- Revise SortTest to be SortComparisonTable.

5. Detailed Hint

- Need to define data fields in classes InsertionSortTest, MergeSortTest, QuickSortTest, SortComparisonTable to generate values of **msMoves, qsCount, isCount**.

6. Submission Requirements & Grading

This assignment is **due by 11:50PM, Thursday, November 12, 2015**.