COMP 3761 Assignment 5

Due: Wednesday **March 04, 2015** at 6:30pm

Note: Test both Problem 1 and 2 with the same input data as you have used in Assignment 3. Download the input file in the a3_input folder.

1. (a) Given a subarray $A[\ell..r]$ of array $A[0..n-1]$, implement the Lomuto's Partition Algorithm by using the **Last** element in the subarray as the pivot value. [4 points]

   (b) Implement a **non-recursive** version of QuickSelect algorithm to find the median of a given array $A[0..n-1]$ using the Lomuto's Partition. [4 points]

   (c) Test your implementation of QuickSelect with the 3 input arrays. Clearly show the median value of the input array in each case. [2 points]

   Note: your program should not sort the entire array in order to report the median in each case.

2. In Assignment 1, you have implemented a brute-force algorithm for counting the number of inversions in a given array with n numbers, which has an $\Theta(n^2)$ time efficiency class.

   (a) Design a **Divide-and-Conquer** algorithm (pseudocode) to count the number of inversions in a given array with n numbers. Analyze the worst-case time efficiency class of your solution. [4 points]

   (b) Implement an **iterative and in-place** version of the divide-and-conquer algorithm of InversionCount. Test your solution in the same way as you have done in Assignment 3. [4 points]

   (c) Compare both the number of inversions and the actual number of basic operations performed in your divide-and-conquer solution with the brute-force version in Assignment 1 for the same input arrays. [2 points]

3. Implement the QuickSort algorithm by three different ways of selecting the pivot in the Hoare's partition process. Given a subarry $A[l..r]$ of the input array $A[0..n-1]$ where $0 \le l \le r \le n-1$, implement the partition function using the following pivot values:

   (a) the **FIRST** element in the subarray

   (b) the **MEDIAN** of the three values at the chosen positions (the first, the middle, and the last index) in the subarray

   (c) a randomly selected element in the subarray.

   Test your implementation of each version of the QuickSort algorithm with the following 3 different types of input data instances of size n, where $n = 100000$:

(1) a randomly generated array of integers in the range of 1 and n.

(2) an array of integers in the increasing order: $1, 2, 3, \ldots, n$.

(3) an array of integers in the decreasing order: $n, n-1, \ldots, 1$.

Your test program must do the following:

- test each version of the QuickSort with all three types of input array values, including the array in type (2) that is already populated in the sorted order.
- verify that the input array is indeed sorted in the increasing order after each QuickSort call.
- Compare the actual basic operations executed in each version for each types of input data. Display your test results. Summarize your observations based on your own test results.

IMPORTANT NOTES:

- You should use a single program (with one main method) to test all the problems in this assignment.
- Design your test program carefully to avoid any (unnecessary) duplicate code.
- As in Assignment 3, you can hard-code the given input file names, but do NOT use any absolute file path in the code. You can simply copy the files to the same directory where your java code is located.
- Print out the required program output only and submit the test results.