University of Windsor

School of Computer Science

60-254 Data Structures and Algorithms – Winter 2017
Lab Assignment 3

Deadline: **February 27/March 1, 2017** (15 minutes prior to the end of the lab)

Regulations: This assignment must be done individually. Any similarity between your code/answers and another student's, or a carbon-copy of an answer found in the web will be considered cheating. None of the built in sorting methods/functions of Java/C/C++ (or other languages) can be used here. You must implement **your own** sorting algorithms.

Objective: The aim of this assignment is to obtain hands on experience in understanding, implementing, testing and comparing sorting algorithms.

Tasks:

1. Design an algorithm in **pseudocode** for in-place Insertion-Sort (implemented on an array), as discussed in class, and which is used to sort integers in **increasing** order.
2. Implement the Insertion-Sort algorithm you designed in #1 in your favorite programming language.
3. Implement the in-place Quicksort algorithm on an array, again, used to sort integers in **increasing** order, where the pivot is always chosen as the **last** element of the list.
4. What are the worst and best-case time complexities in O-notation of
   a. Insertion-Sort?
   b. Quicksort?

   Why? Your answer must be given based on the algorithms you implemented.
5. Write a program that generates an array A of n random integers, and sorts A using Insertion-Sort and Quicksort. Run your program with arrays of size n = 10, 100, 1000, 10000, 100000, 100000. Keep track of the CPU time each algorithm takes to sort each of these arrays. Comment on the running times you obtained, and compare them to the complexities as discussed in class. Hint: place the CPU times you obtained in a table.

Submission:

1. Your assignment must be submitted during the lab session in the section you are registered in. Any submission *on or 15 minutes* prior to the end of the lab session will **not** be accepted and a **zero** mark will be given. Late assignments will be given a **zero** mark. Submissions by email will **not** be accepted.
2. Provide the Insertion Sort algorithm **in pseudocode** (e.g., written on paper or text editor). Code in a programming language like Java, C/C++ or any other will **not** be accepted.
3. Provide the source code for all your programs.
4. Run both Insertion-Sort and Quicksort during the lab and show that they work for various sample inputs, including (more inputs may be included during the evaluation):
   a. 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
   b. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
   c. 1, 2, 4, 5, 3, 7, 8, 10, 11, 9, 6

5. Comment on how the two algorithms behave for the best and worst case running time on the inputs of #5.
6. Explain how your sorting algorithms work. This will be asked when the lab assignment is being submitted and marks will be deducted if not clear how things work.