COMP-2540 Data Structures and Algorithms – Winter 2024
Lab Assignment 3

<u>Deadline:</u> The assignment must be submitted in the lab section students are registered in, on **February 26 or 28, 2024** (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 on the Web will be considered plagiarism. None of the built-in sorting methods/functions of Java/C/C++/Python (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. [4 marks] 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. [2 marks] Implement the Insertion-Sort algorithm you designed in #1 in your favorite programming language.

3. [5 marks] 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. [2 marks] Write an algorithm for Merge-sort that uses queues. Your algorithm must use the queue ADT that you implemented in Assignment 2 (not the actual arrays, but must use the queues). That is, the input to the algorithm is a queue, and when partitioning is done, two queues are created (L and G). Then, L and G are sorted recursively in the same way and merged when sorted. Note that you have already implemented the Merge algorithm that uses queues!

5. [4 marks] Implement your Merge-sort algorithm in your favorite programming language.

6. [3 marks] What are the worst, average and best-case time complexities in O-notation of

   a. Insertion-Sort?

   b. Quicksort?

   c. Merge-sort?

   Why? Your answer must be given based on the algorithms you implemented.

7. [<u>1 bonus mark</u>] Write a program that generates an array A of n random integers, and sorts A using Insertion-Sort, Quicksort and Merge-sort. Run your program with arrays of size n = 64, …, 2^{20}. 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 on a table or a plot (or both) to interpret the results. Note: the array A needs to be converted into a queue before it's input to the Merge-sort algorithm.

8. [<u>1 bonus mark</u>] Implement the recursive algorithm that reverses an array, and run it on the sorted lists of Insertion-Sort, Quicksort and Merge-sort. The output should be lists sorted in **decreasing** order. Note: the output of Merge-sort should be converted from a queue to an array before being reversed.

Total marks: 20 (22 with bonus marks)

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 algorithms.
4. Run Insertion-Sort, Quicksort and Merge-sort 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. 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 your algorithms and programs work.