CSC/DSCI 2720 Data Structure
Lab 4

Due: 02/04/2023 @ 11:59PM

Requirements:

(Failure to follow the requirements will result in a score of Zero.)

1. You may use whatever IDEs / editors you like, but you must submit your responses on iCollege as **.py files**.
2. Your submissions will work exactly as required.
3. Make sure you submit a file that compiles.
4. Your submission will show an output. Should you recive a Zero for no output shown do not bother to email with "but the logic is perfect".
5. Your program's output must exactly match the specs (design, style) given here for each problem to pass the test cases.
6. Design refers to how well your code is written (i.e., is it clear, efficient, and elegant), while Style refers to the readability of your code (correct indentation, good variable names). Add comments to have necessary explanations for your program.
7. Add a "heading" at the very beginning of your .java files as follow:
   Your Name
   CSc 2720 Lab #N
   Lab time: put your lab time here
   Due time: put the due date here

In computer science, a **sorting algorithm** is an algorithm that puts elements of a list into an order.

Below is how the input array is represented LST = [50, 11, 33, 21, 40, 50, 40, 40, 21].

Below is the expected output after sorting LST = [11, 21, 21, 33, 40, 40, 40, 50, 50].

ATTN :

You can use online resource to figure out how a sorting algorithm works. Please be reminded that you cannot use library functions to sort. Doing so will result in a score of Zero!

Exercise 1

(40 points) Write a Java program called ***InsertionSort*** that takes an input array [50, 11, 33, 21, 40, 50, 40, 40, 21] and generates an output array [11, 21, 21, 33, 40, 40, 40, 50, 50]. Users might also give any input arrays. Test your program with different testcases.

Exercise 2

(40 points) Write another Java program called ***SelectionSort*** that takes an input array [50, 11, 33, 21, 40, 50, 40, 40, 21] and generates an output array [11, 21, 21, 33, 40, 40, 40, 50, 50]. Users might also give any input arrays. Test your program with different testcases.

Exercise 3

(20 points) Please do a Big-O analysis for both sorting algorithms. Please analyze both time and space that each algorithm takes. You need to create a new Java file and analyze the program of each algorithm line by line like how we did in the class.

Here is an example for this exercise. In "exercise 4.java" file:

```
//Big-O time analysis for insertion sort
…

//Big-O space analysis for insertion sort
…

//Big-O time analysis for selection sort
…

//Big-O space analysis for selection sort
…
```