ALGORITHMS & DATA STRUCTURES
Lahcen Ouarbya

TOPIC 3: SORTING
LAB. SUBMISSION

In this activity, you will work **individually** implementing sorting algorithms. Remember that you can discuss ideas with your classmates, but you cannot see other's work neither others can see your individual work.

After finishing the implementation of your sorting algorithms, you must upload a file with your code using the "Sorting Algorithms" link available at learn-gold, Week 7.

PART 1: Hybrid Sort (50 points)

Hybrid Sort is an algorithm that mixes the ideas behind Bubble Sort and Selection Sort.

At the end of the first iteration of Bubble Sort (when sorting from smallest to largest), the greatest value is stored in the last position of the array whilst at the end of the first iteration of Selection Sort (when sorting from smallest to largest) the lowest element is stored in the first position of the array.

Hybrid sort will mix both sorting algorithms in such a way that at the end of the first iteration the largest element is in the last position of the array (due to Bubble Sort) and the lowest element will be stored in the first position of the array (due to Selection Sort).

To do so, Hybrid Sorts uses the code of Bubble Sort as the main code and includes part of Selection Sort in it as follows: whilst Bubble Sorts compares pairs of values in the array to decide whether to swap them or not (inner loop), it will use this very same loop to look for the minimum value in the array and store it in the right position at the end of this inner iteration. Once the minimum value is found, it is swapped with the number stored in the position where the minimum value should be stored (that is the Selection Sort part).

The following table shows how the content of an array made of 4 elements ([9,8,7,6]) changes as Hybrid Sort is applied:

[9,8,7,6]
[8,9,7,6] Number 9 is bubbled up, 8 is recorded as the temporal minimum value
[8,7,9,6] Number 9 is bubbled up, 7 is recorded as the temporal minimum value
[8,7,6,9] Number 9 is bubbled up, 6 is recorded as the definite minimum value
[6,7,8,9] Number 6 is swapped with the number in the first position (number 8)

Because at least one swap was made, Hybrid Sort goes through the array again. This time no swap is made (as the array is already sorted) and the algorithm finishes.

PART 2: Mountain Sort (50 points)

Implement Mountain Sort, an algorithm with a **time complexity of Θ(N)** that sorts the integer numbers stored in an array in the following way:

- The first N/2 lowest numbers are stored from smallest to largest in the left half of the array
- The next N/2 numbers are stored from largest to smallest in the right half of the array

For example, if the input array is A=[34, 12, 7, 43, 55, 97, 41, 28, 2, 62], the sorted array should be [**2,7,12,28,34**,97,62,55,43,41]. You may assume that the number of elements in the array is even.
If your algorithm does not have the required time complexity, you will not get any points in this question.

INSTRUCTIONS:

- Download the skeleton of the code for each sorting algorithm and write the code inside the corresponding functions. You cannot change the prototype of the sorting algorithm function, but you can add other functions/libraries if you consider it useful.
- Your code will be tested with "normal" and "extreme" cases (e.g. empty array, one element array, array with the same element in every position, etc). Make sure your code works ok not only for the "normal" case, but also for "extreme" cases.