CSE 101 Summer 2022
Programming Assignment 3

Instructions:
Solutions to this assignment must be submitted through Gradescope by **Sunday, August 21st 2022 at 10 PM**— no late submissions will be accepted.
Please use the submission template provided for this PA to get started.

1. **(Counting Inversions)**
We are given an array L of $n > 0$ distinct integers. We say that two indices $i < j$ form an inversion if $L[i] > L[j]$; that is, if the two elements $L[i]$ and $L[j]$ are "out of order."

In this problem, we seek to count the number of inversions in our array.

Using divide and conquer, write a procedure called `myCount(L)` that takes as input

- `L`: an array of $n > 0$ distinct integers.

and returns a pair $(count, L_{sorted})$ containing

- *count*: the total number of inversions in L.
- L_{sorted}: an array containing every element of L, but sorted.

For example, consider the array

$$L = [6, 1, -4, 10, 2, 7].$$

This array has six inversions: the index pairs $(0,1), (0,2), (0,4), (1,2), (3,4), (3,5)$. Therefore, our algorithm would return the tuple $(6, [-4, 1, 2, 6, 7, 10])$.

Your algorithm should run in $O(n \log n)$ time.

Important Note: You will receive **no points** for this question if found using a brute-force method with a time complexity of $O(n^2)$ or worse. Your program may be manually verified to ensure this.

Hint. Modify the MergeSort algorithm accordingly.

2. **(Closest pair of points)**
   Given an array of n points on a plane $P = [(x_1, y_1), \dots, (x_n, y_n)]$, find the *square* of the distance of the closest pair of points in the array.

   Using divide and conquer, write a procedure called `myMinDistance(P)`. The input format for `myMinDistance` will differ depending on whether C++ or Python is used.

   If you choose to write your algorithm in C++, then the input will be an array P, where

   - P is an array of Points.
   - Points are a class with two defined fields: x and y. The tuple (x, y) represents a point in the Cartesian plane.

   If you choose to write your algorithm in Python, then the input will be an array P, where

   - `P` is an array of $n > 0$ tuples (x_j, y_j) where x_j, y_j are integers. The tuple (x_j, y_j) represents a point in the Cartesian plane.

   For both C++ and Python, we return

   - *min*: the square of the minimum distance of points in P.

   For example, consider the array

   $$P = [(0, 1), (-5, 3), (4, 2), (2, 0)].$$

   The two closest points in this array are $(0, 1)$ and $(2, 0)$. The square of their distance is 5, so our algorithm would return 5.

   Your algorithm should have $O(n(\log n)^2)$ runtime or better.

   Important Note: You will receive **no points** for this question if found using a brute-force method with a time complexity of $O(n^2)$ or worse. Your program may be manually verified to ensure this.