CS 577: Introduction to Algorithms	**Program 2 - Crossing Lines**
Out: February 9, 2021	Due: February 16, 2021

Coding Question:

Reminders:

- Must be coded individually in your choice of either Python, Java, C, C++, or C#
- There are hidden testcases
- Submitted through Gradescope
- There is a class-wide runtime leaderboard on Gradescope
- We encourage the use of Piazza for debugging help
- Please do not cheat

Problem:

Suppose you are given two sets of n points, one set $\{p_1, p_2, \ldots, p_n\}$ on the line $y = 0$ and the other set $\{q_1, q_2, \ldots, q_n\}$ on the line $y = 1$. Create a set of n line segments by connecting each point p_i to the corresponding point q_i. Your goal is to develop an algorithm to determine how many pairs of these line segments intersect. Your algorithm should take the $2n$ points as input, and return the number of intersections. Using divide-and-conquer, you should be able to develop an algorithm that runs in $O(n \log n)$ time.

Hint: What does this problem have in common with the problem of counting inversions in a list?

Input should be read in from stdin. The first line will contain the number of pairs of points (n). The next n lines each contain the location of a point q_i on the top line. The final n lines each contain the location of the corresponding point p_i on the bottom line. For the example shown in Fig 1, the input is properly formatted in the first test case below.

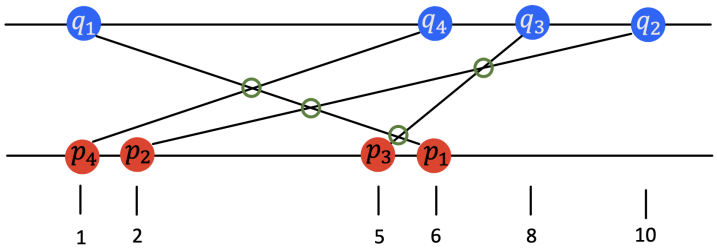

Figure 1: An example for the line intersection problem where the answer is 4

Constraints:

- $1 \leq n \leq 10^6$
- For each point, its location x is a positive integer such that $1 \leq x \leq 10^6$
- No two points are placed at the same location on the top line, and no two points are placed at the same location on the bottom line.
- Note that in C\C++, the results of some of the test cases may not fit in a 32-bit integer. If you are using C\C++, make sure you use a 'long long' to store your final answer.

Sample Test Cases:

input:
4
1
10
8
6
6
2
5
1

expected output:
4

input:
5
9
21
1
5
18
2
4
6
10
1

expected output:
7