Problem Statement:

[Part B]: Divide and Conquer Programming (22%)

1. [12%] Implement the algorithm for finding the closest pair of points in two dimension plane using divide and conquer strategy.

A system for controlling air or sea traffic might need to know which are the two closest vehicles in order to detect potential collisions. This part solves the problem of finding the closest pair of points in a set of points. The set consists of points in R^2 defined by both an x and a y coordinate. The "closest pair" refers to the pair of points in the set that has the smallest Euclidean distance, where Euclidean distance between points $p_1=(x_1,y_1)$ and $p_2=(x_2,y_2)$ is simply $\text{sqrt}((x_1-x_2)^2+(y_1-y_2)^2)$. If there are two identical points in the set, then the closest pair distance in the set will obviously be zero.

Input data: n points with coordinates:

X coordinates: p[0].x, p[1].x, p[2].x,…., p[n].x

Y coordinates: p[0].y, p[1].y, p[2].y,…., p[n].y

Output: minimum distance between points p[i] and p[j] (index i and j should be identified)

Input data for test:

```
n = 10000;
for(i=0; i<n; i++)
 {
  p[i].x= n- i;
  p[i].y= n- i;
   }
```

Other Input data for test:

```
n = 10000;
for(i=0; i<n; i++)
 {
  p[i].x= i*i;
  p[i].y= i*i;
   }
```

Or:

Input:

If X>0
X=1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, ……,19995, 19997, 19999 (odd numbers)
Y=int($\sqrt{9999^2 - (X - 10000)^2}$)

Else
X=0, -2, -4, -6, -8, -10, -12, ……, -19996, -19998, -20000 (even number)
Y= int($\sqrt{10000^2 - (X + 10000)^2}$)

2. (10%) Use the brute-force approach to solve the above problem. Compare the two implemented algorithms in terms of time complexity. Print out the time cost during the execution of these two algorithms.

3. (Optional: Extra 10%) apply the close-pair algorithm to the three dimensional case.