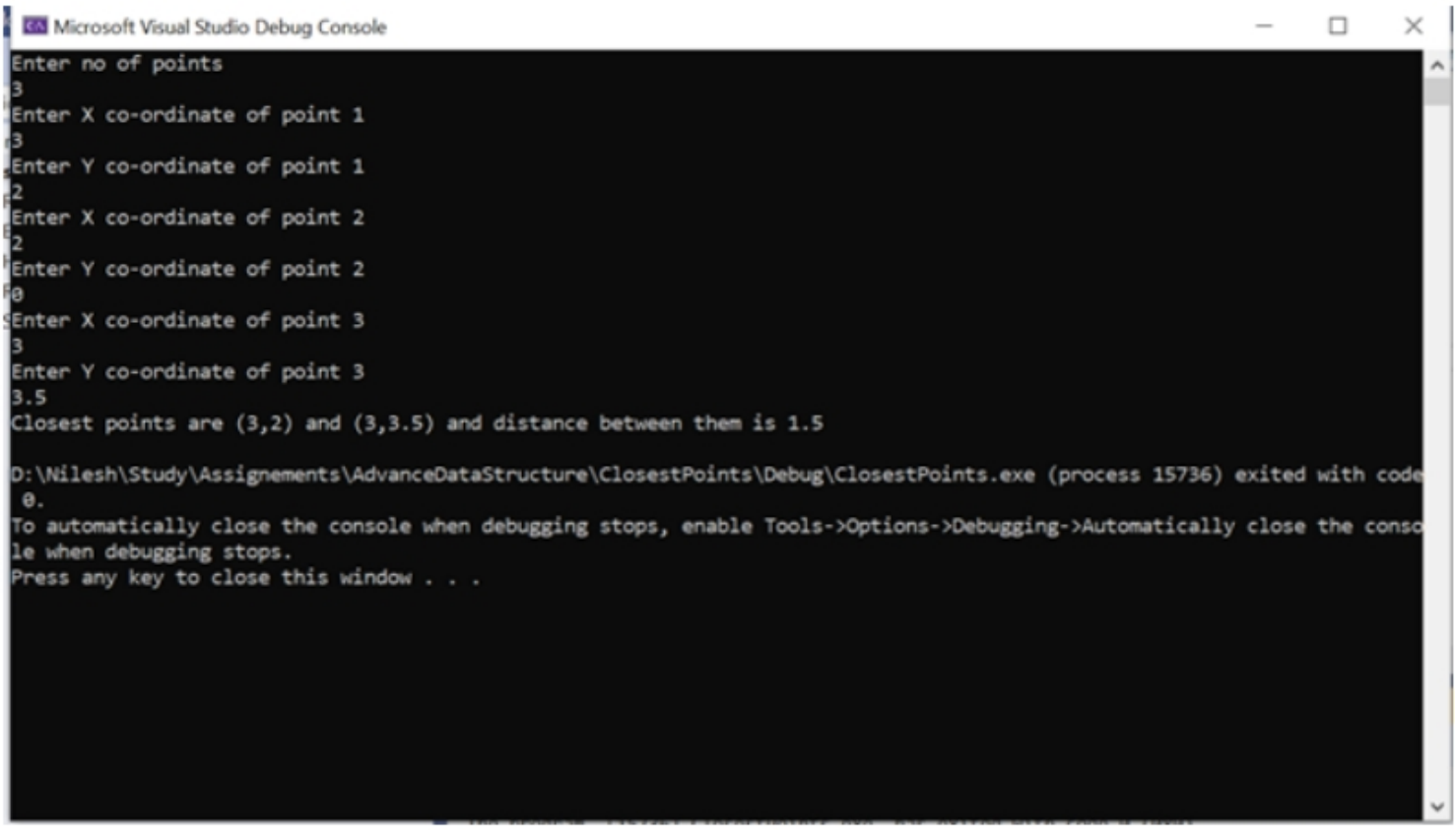
Name- Nilesh Shivanand Kale
Roll No - CS21MTECH11022
Subject – Advance Data Structre andAlgorithms
Assignement -Closest point problem

Snapshots of output window:

Test case 1:

Points [(3,2), (2,0), (3, 3.5)]

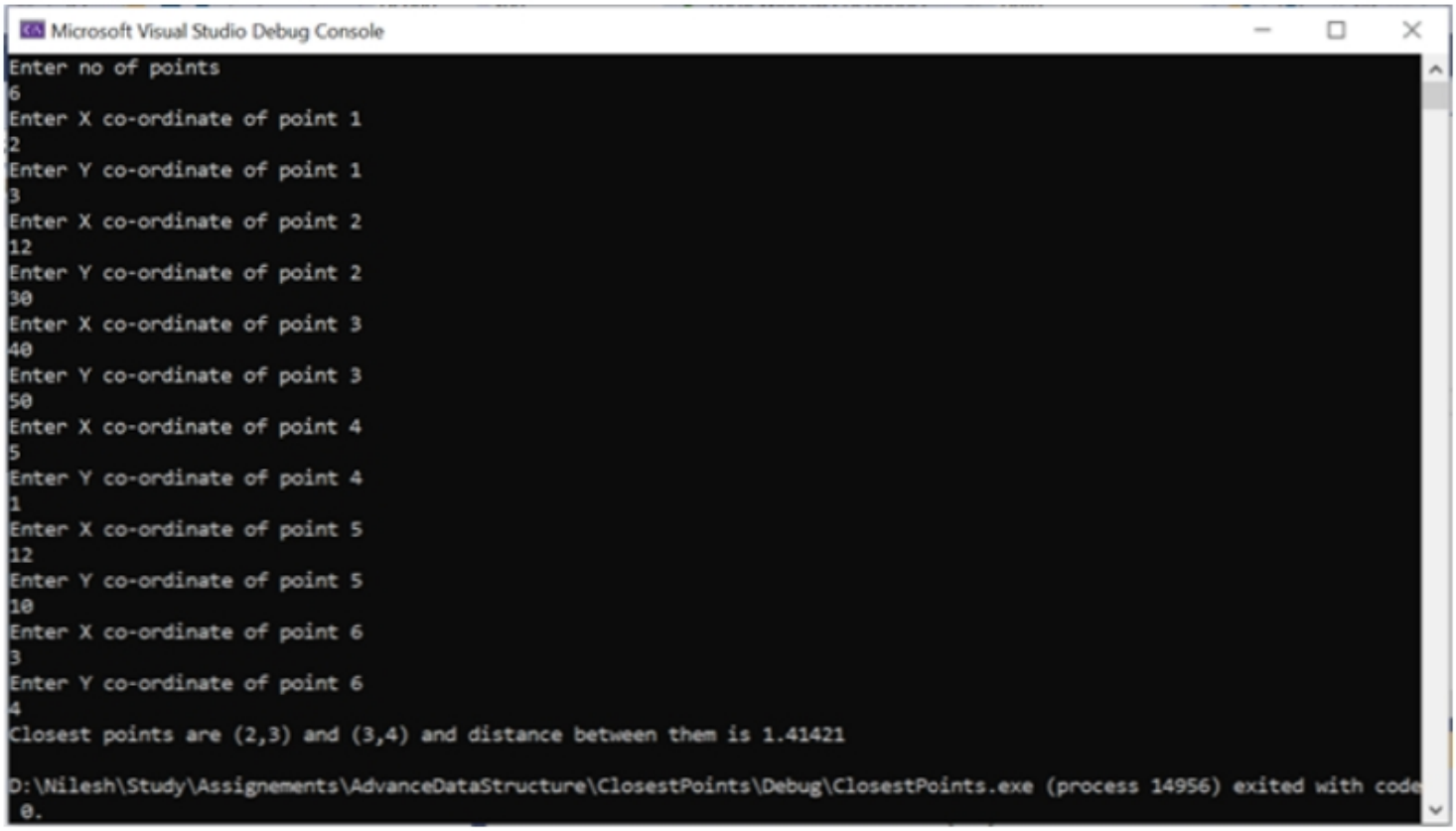
Output - Closest points are (3,2) and (3,3.5) and distance between them is 1.5



Test case 2:

Points[(2,3), (12,30), (40,50), (5,1), (12,10), (3,4)]

Output - Closest points are (2,3) and (3,4) and distance between them is 1.41421



Algorithm:

```
Main():
    Vector  Major_X_CoordinatesVector ,  Major_y_CoordinatesVector
    Vector  Minor_X_CoordinatesVector ,  Minor_y_CoordinatesVector

    for i=1 to no_of_points
        input x,y co-ordinates

    # sort major vectors pair by
    x-coordinate and minor vectors pair by y-coordinate

    MergeSort(Major_X_CoordinatesVector,  Major_y_CoordinatesVector)        ..... o(n logn)
    MergeSort(Minor_Y_CoordinatesVector,  Minor_X_CoordinatesVector)        ..... o(n logn)

    FindClosestPair(Majorvectors, Minor vectors,  low, high)
```

```
FindClosestPair(Major vectors, Minor vectors, low, high):                   ..... T(n)

    no_of_points= high – low + 1

    #Base case
    if no_of_points<= 3 : return BruteForceCalculation(Major vectors, Minor vectors,low, high)

    #devide
    Mid = ( high –low ) / 2

    D1 = FindClosestPair(Majorvectors, Minor vectors, low, mid)
    D2 = FindClosestPair(Majorvectors, Minor vectors, mid+1, high)          ..... T(n/2)

    D = Find_Minimum_Of(D1, D2)                                             ..... T(n/2)

    # Find minimum in crossingstrip

    # Create strip vector from minor vector(sorted by y-coordinate)containspoints
    # having x-coordinate at max distance D from mid

    for i=0 to no_of_points
        if Minor_X_vector[i]– mid < d
            add point in strip vector

    strip_d = FindClosestPointInCrossing(StripVector,  D)                   ..... o(n)

    if strip_d < D                                                          ..... o(n)
        retrun strip_d
    else
        return D
```

```
FindClosestPointInCrossing(Strip  Vector, D):

    # As stripcontain point  in ascending order of ycoordinate

    MinimumDistance = D
    for i=0 tostrip_size
        for j=i+1 to strip_size
            MinimumDistance = Distance( point i,  point j)
            if  MinimumDistance  < D                                        ..... o(n)
                D= MinimumDistance

    return D
```

Running time complexity:

T(n) is time complexity of FindClosestPoinnts()

=>T(n) = 2T(n/2) + O(n) + O(n)

Strip creation

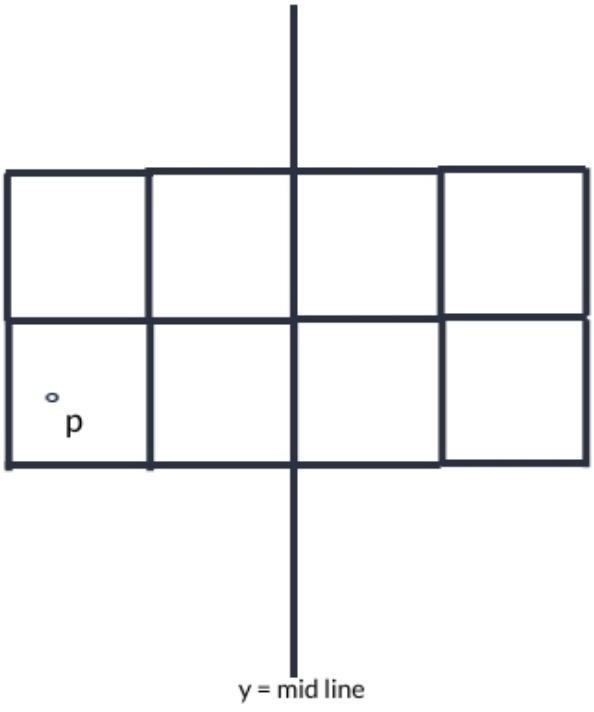
FindClosestPointInCrossing(Strip Vector, D):

Using Master's theorem,

=> T(n) ∈ O(n logn)

Key point:

Why FindClosestPointInCrossing(Strip Vector, D) runs in O(n) even it look like O(n)



Let p is point located at distance less than d from y=mid.

Each square is of length d/2 .

We want to make sure that whatever points selected in strip vector should not belong to same box shown.

Proof by contradiction:
Assume there are two points in a square of d/2 * d/2.
Then maximum distance between them can be d/(2^0.5) which will be less than d and it must greater than d by algorithm design.
So, here our assumption failed.
and Hence the proof that, each square contain max 1 point.

So, point P need to check with only with other 7 points which have higher y-coordinate.
Hence, inner loop in function will run for only 7 times , not for n time and hence we can say,

FindClosestPointInCrossing(Strip Vector, D) ∈ O(n)

Prooof of correctness:

In algorithm FindClosestPoinnts(),

Base case:

if no_of_point <= 3, We are finding it by Brute force algorithm. So, base case is verified.

Induction step:

FindClosestPoinnts(n) = FindClosestPoinnts(n/2) + FindClosestPoinnts(n/2) + FindClosestPointInCrossing(k):
where k <= n and running in brute fore algorithm

So, induction step is verified.

Hence the proof that, our algorithm is correct.