**EXPLANATION :**

**1)** We sort all points according to x coordinates.

**2)**Divide all points in two halves.

**3)** Recursively finds the smallest distances in both sub arrays

**4)**Take the minimum of two smallest distances. Let the minimum be d.

**5)** Create an array strip[] that stores all points which are at most d distance away from the middle line dividing the two sets.

**6)**Find the smallest distance in strip[].

**7)** Return the minimum of d and the smallest distance calculated in above step 6.

**CODING**

**#include <iostream.h>**

**#include <float.h>**

**#include <stdlib.h>**

**#include <math.h>**

**using namespace std;**

**class Point**

**{**

**Public:**

**int x,y;**

**};**

**Int compareX(const void\* a, const void\* b)**

**{**

**Point \*pl=(point \*a)a,\*p2=(point \*b);**

**return ( p1->y – p2->y);**

**}**

**int compareY (const void\* a, const void\* b)**

**{**

**Point \*p1 = (Point \*) a, \*p2 = (Point \*) b;**

**return (p1->x - p2->x);**

**}**

**int compareY(const void\* a, const void\* b)**

**{**

**Point \*p1 = (Point \*)a, \*p2 = (Point \*)b;**

**return (p1->y - p2->y);**

**}**

**float dist(Point p1, Point p2)**

**{**

**return sqrt( (p1.x - p2.x)\*(p1.x - p2.x) +**

**(p1.y - p2.y)\*(p1.y - p2.y)**

**);**

**}**

**float brute Force(Point P[], int n)**

**{**

**float min = FLT\_MAX;**

**for (int i = 0; i < n; ++i)**

**for (int j = i+1; j < n; ++j)**

**if (dist(P[i], P[j]) < min)**

**min = dist(P[i], P[j]);**

**return min;**

**}**

**float min(float x, float y)**

**{**

**return (x < y)? x : y;**

**}**

**float stripClosest(Point strip[], int size, float d)**

**{**

**float min = d; // Initialize the minimum distance as d**

**for (int i = 0; i < size; ++i)**

**for (int j = i+1; j < size && (strip[j].y - strip[i].y) < min; ++j)**

**if (dist(strip[i],strip[j]) < min)**

**min = dist(strip[i], strip[j]);**

**return min;**

**}**

**float closestUtil(Point Px[], Point Py[], int n)**

**{**

**if (n <= 3)**

**return brute Force(Px, n);**

**int mid = n/2;**

**Point midPoint = Px[mid];**

**Point Pyl[mid];**

**Point Pyr[n-mid];**

**int li = 0, ri = 0;**

**for (int i = 0; i < n; i++)**

**{**

**if (Py[i].x <= midPoint.x && li<mid)**

**Pyl[li++] = Py[i];**

**else**

**Pyr[ri++] = Py[i];**

**}**

**float dl = closestUtil(Px, Pyl, mid);**

**float dr = closestUtil(Px + mid, Pyr, n-mid);**

**float d = min(dl, dr);**

**Point strip[n];**

**int j = 0;**

**for (int i = 0; i < n; i++)**

**if (abs(Py[i].x - midPoint.x) < d)**

**strip[j] = Py[i], j++;**

**return stripClosest(strip, j, d);**

**}**

**float closest(Point P[], int n)**

**{**

**Point Px[n];**

**Point Py[n];**

**for (int i = 0; i < n; i++)**

**{**

**Px[i] = P[i];**

**Py[i] = P[i];**

**}**

**qsort(Px, n, sizeof(Point), compareX);**

**qsort(Py, n, sizeof(Point), compareY);**

**return closestUtil(Px, Py, n);**

**}**

**int main()**

**{**

**Point P[] = {{2, 3}, {12, 30}, {40, 50}, {5, 1}, {12, 10}, {3, 4}};**

**int n = sizeof(P) / sizeof(P[0]);**

**cout << "The smallest distance is " << closest(P, n);**

**return 0;**

**}**

**OUTPUT**

