

```

1  /***** point on ploygon *****/
2  #include <iostream>
3  #include <string>
4  #include <vector>
5  #include <iterator>
6  #include <algorithm>
7  #include <cstdio>
8  using namespace std;
9
10 // NB: max_int would cause overflow in the
orientation test
11 const int INF = 1000002;
12
13 struct Point {
14     int x, y;
15 };
16
17 ostream& operator<<(ostream& os, const Point& p) {
18     os << "{" << p.x << " " << p.y << "}" ;
19     return os;
20 }
21 ostream& operator<<(ostream& os, const
vector<Point>& p) {
22     os << "{" ;
23     copy(p.begin(), p.end(),
ostream_iterator<Point>(os));
24     os << "}" ;
25     return os;
26 }
27
28 int orientation(const Point& p1, const Point& p2,
const Point& q1) {
29     int val = (p2.y - p1.y) * (q1.x - p2.x) - (q1.y -
p2.y) * (p2.x - p1.x);
30     if (val == 0)
31         return 0;
32     else
33         return (val < 0) ? -1 : 1;
34 }
35
36 // Returns true if q lies on p1-p2
37 bool onSegment(const Point& p1, const Point& p2,
const Point& q) {
38     if (min(p1.x, p2.x) <= q.x && q.x <= max(p1.x,
p2.x)
39         && min(p1.y, p2.y) <= q.y && q.y <= max(p1.y,
p2.y))
40         return true;
41     else
42         return false;
43 }
44
45 bool intersectionTest(const Point& p1, const Point&
p2,
46 const Point& p3, const Point& p4) {
47     int o1 = orientation(p1, p2, p3);
48     int o2 = orientation(p1, p2, p4);
49     int o3 = orientation(p3, p4, p1);
50     int o4 = orientation(p3, p4, p2);
51
52     // General case
53     if (o1 != o2 && o3 != o4)
54         return true;
55
56     // Special cases
57     if (o1 == 0 && onSegment(p1, p2, p3))

```

```

58     return true;
59     if (o2 == 0 && onSegment(p1, p2, p4))
60         return true;
61     if (o3 == 0 && onSegment(p3, p4, p1))
62         return true;
63     if (o4 == 0 && onSegment(p3, p4, p2))
64         return true;
65
66     return false;
67 }
68
69 bool pointInPolygon(const Point& p, const
vector<Point>& polygon) {
70
71     if (polygon.size() < 3)
72         return false; // Flawed polygon
73
74     Point PtoInfinity = { INF , p.y };
75
76     int intersectionsCount = 0;
77     int i = 0, j = i + 1;
78     do {
79
80         if (intersectionTest(p, PtoInfinity,
polygon[i], polygon[j]) == true) {
81
82             ++intersectionsCount;
83
84             if (orientation(polygon[i], polygon[j], p) ==
0) { // Collinear
85                 if (onSegment(polygon[i], polygon[j], p) ==
true)
86                     return true;
87                 else {
88                     // Exception case when point is collinear
89                     // but not on segment
90                     // e.g.
91                     //
92                     //          * *****
93                     //          k /          \ w
94                     //          // The collinear segment is worth 0 if k
95                     //          // and w have the same
96                     //          // vertical direction
97                     int k = (((i - 1) >= 0) ? // Negative
wraparound
98                         (i - 1) %
static_cast<int>(polygon.size())) :
99                         static_cast<int>(polygon.size()) + (i -
1));
100                     int w = ((j + 1) % polygon.size());
101                     if ((polygon[k].y <= polygon[i].y &&
polygon[w].y <= polygon[j].y)
102                         || (polygon[k].y >= polygon[i].y &&
polygon[w].y >= polygon[j].y))
103                         --intersectionsCount;
104                 }
105             }
106         }
107
108         i = (++i % polygon.size());
109         j = (++j % polygon.size());
110     } while (i != 0);
111
112

```

```

113     return (intersectionsCount % 2 != 0);
114 }
115
116 int main() {
117
118     int a,b,c,d,e,i,j,p,q;
119     bool v;
120     Point z;
121
122     vector<Point> polygon;
123     cin>>a;
124
125     for(i=1;i<=a;i++)
126     {
127         cout<<"Case " <<i<<" : " <<endl;
128         polygon.clear();
129         scanf("%d",&b);
130         for(j=0;j<b;j++)
131         {
132             scanf("%d %d",&z.x,&z.y);
133             polygon.push_back(z);
134         }
135         scanf("%d",&c);
136         for(j=0;j<c;j++)
137         {
138             scanf("%d %d",&z.x,&z.y);
139             v=pointInPolygon(z,polygon);
140             if(v==true)
141                 cout<<"Yes" <<endl;
142             else
143                 cout<<"No" <<endl;
144         }
145     }
146
147
148     return 0;
149 }
150

```