

```
1.  /***** Monotone Convex Hull Algo by Nafee *****/
2.  struct point
3.  {
4.      long long x, y;
5.      point(){}
6.      point(long long x, long long y)
7.      {
8.          this->x = x;
9.          this->y = y;
10.     }
11.     point operator - (point B)
12.     {
13.         point ret;
14.         ret.x = x - B.x;
15.         ret.y = y - B.y;
16.         return ret;
17.     }
18.     const bool operator < (point B) const
19.     {
20.         if ( x == B.x )
21.         {
22.             return y < B.y;
23.         }
24.         return x < B.x;
25.     }
26. };
27. bool cmpPoint(point A, point B)
28. {
29.     if ( A.x == B.x )
30.     {
31.         return A.y < B.y;
32.     }
33.     return A.x < B.x;
34. }
35. long long crossPr(point A, point B)
36. {
37.     long long ret = A.x*B.y - A.y*B.x;
38.     return ret;
39. }
40. vector<point> findConHull(vector<point> givenPoints)
41. {
42.     long long a, b, c, d, e, f, t, len = givenPoints.size(), siz;
43.     vector<point> ret;
44.     sort(givenPoints.begin(), givenPoints.end());
45.     for (a = 0; a < len; a++)
46.     {
47.         siz = ret.size();
48.         while( siz >= 2 && crossPr(ret[siz-1]-ret[siz-2], givenPoints[a]-ret[siz-2]) <= 0 )
49.         {
50.             ret.pop_back();
51.             siz = ret.size();
52.         }
53.         ret.push_back( givenPoints[a] );
54.     }
55.     ret.pop_back();
56.     t = ret.size();
57.     len = givenPoints.size();
58.     for (a = len-1; a >= 0; a--)
```

```
59.     {
60.         siz = ret.size();
61.         while( siz >= 2+t && crossPr(ret[siz-1]-ret[siz-2], givenPoints[a]-ret[siz-2]) <= 0 )
62.         {
63.             ret.pop_back();
64.             siz = ret.size();
65.         }
66.         ret.push_back( givenPoints[a] );
67.     }
68.     ret.pop_back();
69.     return ret;
70. }
```