```cpp
#include<bits/stdc++.h>

using namespace std;

struct Point{
    double x,y;
    Point(double x=0,double y=0):x(x),y(y){}
};

typedef Point Vector;

struct Line{
    Point p;
    Vector v;
    double ang;
    Line(){}
    Line(Point p,Vector v):p(p),v(v){ ang = atan2(v.y,v.x); }
    bool operator < (const Line &L) const {
        return ang < L.ang;
    }
};


const double eps = 1e-10;
const double PI = acos(-1);

Vector operator + (Vector A,Vector B){ return Vector(A.x+B.x,A.y+B.y); }
Vector operator - (Point A,Point B){ return Vector(A.x-B.x,A.y-B.y); }
Vector operator * (Vector A,double p){ return Vector(A.x*p,A.y*p); }
Vector operator / (Vector A,double p){ return Vector(A.x/p,A.y/p); }

bool operator < (const Point &a,const Point &b){
    return a.x < b.x || (a.x == b.x && a.y < b.y);
}

int dcmp(double x){
    if( fabs(x) < eps )  return 0;
    else return x < 0 ? -1 : 1;
}

bool operator == (const Point &a,const Point &b){
    return dcmp(a.x-b.x) == 0 && dcmp(a.y-b.y);
}

double Dot(Vector A,Vector B){ return A.x*B.x + A.y*B.y; }
double Length(Vector A){ return sqrt(Dot(A,A)); }
double Angle(Vector A,Vector B){ return acos( Dot(A,B)/Length(A)/Length(B)
); }

double Cross(Vector A,Vector B){ return A.x*B.y - A.y*B.x; }
double Area2(Point A,Point B,Point C){ return Cross(B-A,C-A); }

Vector Rotate(Vector A,double rad){
    return Vector( A.x*cos(rad)-A.y*sin(rad),A.x*sin(rad)+A.y*cos(rad) );
}
```

```
55
56  bool OnSegment(Point p,Point a1,Point a2){
57      return dcmp(Cross(a1-p,a2-p)) == 0 && dcmp(Dot(a1-p,a2-p)) < 0;
58  }
59
60  //  点P在有向线段左边
61  bool OnLeft(Line L,Point P){
62      return Cross(L.v,P-L.p) > 0;
63  }
64
65  //  两直线交点
66  Point GetIntersection(Line a,Line b){
67      Vector u = a.p - b.p;
68      double t = Cross(b.v,u) / Cross(a.v,b.v);
69      return a.p+a.v*t;
70  }
71
72  double ConvexPolygonArea(Point *P,int n){
73      double area = 0;
74      for(int i=1;i<n-1;i++)
75          area += Cross(P[i]-P[0],P[i+1]-P[0]);
76      return area/2;
77  }
78
79  //  半平面交
80  double HalfplaneIntersection(Line *L,int n,Point *poly){
81      sort(L,L+n);
82      int first,last;
83      Line *q = new Line[n];    //双端队列
84      Point *p = new Point[n];  //p[i] 为 q[i] 和 q[i+1] 的交点
85      q[first = last = 0] = L[0];
86      for(int i=1;i<n;i++)
87      {
88          while( first < last && !OnLeft(L[i],p[last-1]) )  last--;
89          while( first < last && !OnLeft(L[i],p[first]) )  first++;
90          q[++last] = L[i];
91          // 平行取内侧
92          if( fabs( Cross(q[last].v,q[last-1].v) ) < eps )
93          {
94              last--;
95              if( OnLeft(q[last],L[i].p) )  q[last] = L[i];
96          }
97          if( first < last )  p[last-1] = GetIntersection(q[last-
    1],q[last]);
98      }
99      //  删除无用平面
100     while( first < last && !OnLeft(q[first],p[last-1]) )  last--;
101     if( last - first <= 1 )  return 0;
102     p[last] = GetIntersection(q[last],q[first]);
103
104     int m = 0;
105     for(int i=first;i<=last;i++)
106         poly[m++] = p[i];
107     return ConvexPolygonArea(poly,m);
108 }
109
110 int tol;
111 Point PP[550];
```

```
Line LL[5050];

int main(int argc,char ** argv){
    int T,n;
    scanf("%d",&T);
    while( T-- )
    {
        scanf("%d",&n);
        scanf("%lf %lf",&PP[0].x,&PP[0].y);
        PP[n] = PP[0];
        for(int i=1;i<n;i++)
            scanf("%lf %lf",&PP[i].x,&PP[i].y);
        for(int i=0;i<n;i++)
            LL[tol++] = Line(PP[i],PP[i+1] - PP[i]);
    }
    printf("%.3lf",HalfplaneIntersection(LL,tol,PP));
    return 0;
}

/*

Luogu
P4196 [CQOI2006]凸多边形

逆时针给出 n个点的坐标 所构成的 T 个凸多边形 求面积并
对一个凸多边形上的点 逆时针地将相邻构成线段(使得左边为多边形的区域)
将线段求 半平面交
最后将所给的点集求一次凸多边形的面积即可

*/
```