// 计算几何大全套

const int MAXN=1e5;

inline double myabs(double x) {if (x>eps) return x;if (x<-eps) return -x;return 0;}

inline bool islarger(double x,double y) {return x-y>eps;}

inline bool issmaller(double x,double y) {return x-y<-eps;}

inline bool isequal(double x,double y) {return myabs(x-y)<eps;}

struct Vector

{

double x,y;

Vector () {}

inline Vector (double xx,double yy) {x=xx;y=yy;}

inline double calc\_k()

{

if (isequal(x,0)) return islarger(y,0)?INF:-INF;

return y/x;

}

inline double calc\_ang() {return atan2(y,x);}

inline Vector operator + (Vector other) {return Vector(x+other.x,y+other.y);}

inline Vector operator - (Vector other) {return Vector(x-other.x,y-other.y);}

inline Vector operator \* (double t) {return Vector(x\*t,y\*t);}

inline double dot(Vector other) {return x\*other.x+y\*other.y;}

inline double det(Vector other) {return x\*other.y-y\*other.x;}

};

struct Seg

{

Vector st,ed;

Seg () {}

inline Seg(Vector ss,Vector ee) {st=ss;ed=ee;}

inline LB dot(Seg other) {return (ed-st).dot(other.ed-other.st);}

inline LB det(Seg other) {return (ed-st).det(other.ed-other.st);}

};

struct Line

{

Vector pt1,pt2;int ind;

double ang;

inline void set\_ang() {ang=(pt2-pt1).calc\_ang();}

inline bool operator < (const Line &other) const {return ang<other.ang;}

inline double dot(Vector other) {return (pt2-pt1).dot(other);}

inline double dot(Line other) {return (pt2-pt1).dot(other.pt2-other.pt1);}

inline double det(Vector other) {return (pt2-pt1).det(other);}

inline double det(Line other) {return (pt2-pt1).det(other.pt2-other.pt1);}

}L[MAXN+48];

inline LB getdist(Vector pt1,Vector pt2) {return sqrt((pt2.x-pt1.x)\*(pt2.x-pt1.x)+(pt2.y-pt1.y)\*(pt2.y-pt1.y));}

inline bool OnTheLeft(Line l,Vector pt) {return islarger(l.det(pt-l.pt1),0);}

inline bool OnTheLine(Line l,Vector pt) {return isequal((l.pt1-pt).det(l.pt2-pt),0) && !islarger((l.pt1-pt).dot(l.pt2-pt),0);}

inline bool on\_seg(Seg s,Vector pt) {return (myabs((s.st-pt).det(s.ed-pt))<eps && (s.st-pt).dot(s.ed-pt)<eps);}

inline Vector intersection(Seg s1,Seg s2) {return s1.st+(s1.ed-s1.st)\*((s2.ed-s2.st).det(s2.st-s1.st)/s2.det(s1));}

inline bool isintersect(Seg s1,Seg s2)

{

if (s1.det(s2)<eps)

{

if (on\_seg(s1,s2.st) || on\_seg(s1,s2.ed) || on\_seg(s2,s1.st) || on\_seg(s2,s1.ed))

return true;

else return false;

}

Vector pt=intersection(s1,s2);

if (on\_seg(s1,pt) && on\_seg(s2,pt)) return true; else return false;

}

inline bool isparallel(Line l1,Line l2) {return isequal(l1.det(l2),0);}

inline Vector intersection(Line l1,Line l2)

{

if (isparallel(l1,l2)) return OnTheLine(l2,l1.pt1)?l1.pt1:l1.pt2;

return l1.pt1+(l1.pt2-l1.pt1)\*((l2.pt2-l2.pt1).det(l2.pt1-l1.pt1)/((l2.pt2-l2.pt1).det(l1.pt2-l1.pt1)));

}

// 半平面交

Line q[MAXN+48];int head,tail;

Vector a[MAXN+48];

inline void HalfPlaneIntersection()

{

sort(L+1,L+n+1);

head=tail=1;q[1]=L[1];

for (register int i=2;i<=n;i++)

{

while (head<tail && !OnTheLeft(L[i],a[tail-1])) tail--;

while (head<tail && !OnTheLeft(L[i],a[head])) head++;

q[++tail]=L[i];

if (head<tail && isparallel(q[tail-1],q[tail]))

{

if (OnTheLeft(q[tail-1],q[tail].pt1)) q[tail-1]=q[tail];

tail--;

}

if (head<tail) a[tail-1]=intersection(q[tail-1],q[tail]);

}

while (head+1<tail && !OnTheLeft(q[head],a[tail-1])) tail--;

}

// 求凸包

inline int ConvexHull()

{

sort(cpt+1,cpt+tot+1);int cnt=0;

for (register int i=1;i<=tot;i++)

{

while (cnt>=2 && !islarger((conv[cnt]-conv[cnt-1]).det(cpt[i]-conv[cnt]),0)) cnt--;

conv[++cnt]=cpt[i];

}

int k=cnt;

for (register int i=n-1;i>=1;i--)

{

while (cnt>=k+1 && !islarger((conv[cnt]-conv[cnt-1]).det(cpt[i]-conv[cnt]),0)) cnt--;

conv[++cnt]=cpt[i];

}

if (tot>1) cnt--;

return cnt;

}