```
----------------------几何---------------------
//KDtree
const int MAXN=200005,MAXD=2;
const double INF=1e10;
int n,m,D;
struct P{
        int d[MAXD],M[MAXD],m[MAXD],l,r;
        P(int x=0,int y=0) { l=r=0; d[0]=x; d[1]=y; }
        int& operator [](int x){ return d[x]; }
        friend bool operator <(P a,P b){ return a[D]<b[D]; }
        friend bool operator ==(P a,P b){ return
a.d[0]==b.d[0]&&a.d[1]==b.d[1]; }
}p[MAXN];
inline double dis(P a,P b)
{ return
sqrt(1.0*(a[0]-b[0])*(a[0]-b[0])+1.0*(a[1]-b[1])*(a[1]-b[1])); }
inline bool in(int x1,int y1,int x2,int y2,int X1,int Y1,int X2,int Y2)
{ return x1<=X1&&X2<=x2&&y1<=Y1&&Y2<=y2; }
inline bool out(int x1,int y1,int x2,int y2,int X1,int Y1,int X2,int Y2)
{ return x1>X2||x2<X1||y1>Y2||y2<Y1; }
struct KDtree{
        P t[MAXN],T;
        int root,n;
        double ans;
        inline void update(int k){
                int i,l=t[k].l,r=t[k].r;
                for (i=0;i<MAXD;++i) {
                        if (l) {

        t[k].M[i]=max(t[k].M[i],t[l].M[i]);

        t[k].m[i]=min(t[k].m[i],t[l].m[i]);
                        }
                        if (r) {

        t[k].M[i]=max(t[k].M[i],t[r].M[i]);

        t[k].m[i]=min(t[k].m[i],t[r].m[i]);
                        }
                }
        }
        int build(int l,int r,int now){
                D=now;
                int i,mid=(l+r)>>1;
                nth_element(p+l,p+mid,p+r+1); t[mid]=p[mid];
                for (i=0;i<MAXD;++i)
t[mid].M[i]=t[mid].m[i]=t[mid][i];
                if (l<mid) t[mid].l=build(l,mid-1,now^1);
                if (mid<r) t[mid].r=build(mid+1,r,now^1);
                update(mid); return mid;
        }
        inline void build(int nn){ n=nn; root=build(1,nn,0); }
        void insert(int &k,int now){
                if (k==0) {
                        t[k].r=++n; t[n]=T;
                        for (int i=0;i<MAXD;++i)
t[n].M[i]=t[n].m[i]=t[n][i];
                        return;
                }
                if (T[now]>=t[k][now]) insert(t[k].r,now^1);
                else insert(t[k].l,now^1);
                update(k);
        }
        inline void insert(P p){ T=p; T.l=T.r=0; insert(root,0); }
        inline double get(int k,P p){
                double tmp[2]={0.0,0.0};
                for (int i=0;i<MAXD;++i) {
                        if (tmp[i]<t[k].m[i]-p[i])
tmp[i]=t[k].m[i]-p[i];
                        if (tmp[i]<p[i]-t[k].M[i])
tmp[i]=p[i]-t[k].M[i];
                }
                return sqrt(tmp[0]*tmp[0]+tmp[1]*tmp[1]);
        }
        void query(int k,int now){
                double d,dl=INF,dr=INF;
                d=dis(t[k],T); ans=min(ans,d);
                if (t[k].l) dl=get(t[k].l,T);
                if (t[k].r) dr=get(t[k].r,T);
                if (dl<dr) {
                        if (dl<ans) query(t[k].l,now^1);
                        if (dr<ans) query(t[k].r,now^1);
                }
                else {
                        if (dr<ans) query(t[k].r,now^1);
                        if (dl<ans) query(t[k].l,now^1);
                }
        }
        inline double query(P p){ ans=INF; T=p; query(root,0); return
ans; }
}kd;


//Line intersect
inline bool intersect(P a,P b,P c,P d){
        if (max(a.x,b.x)<min(c.x,d.x)||min(a.x,b.x)>max(c.x,d.x))
return false;
        if (max(a.y,b.y)<min(c.y,d.y)||min(a.y,b.y)>max(c.y,d.y))
return false;
        double
x=cross(a-b,c-b),y=cross(a-b,d-b),z=cross(c-d,a-d),w=cross(c-d,b-d);
        if (sgn(x)==0&&sgn(y)==0) return false;
        if (sgn(z)==0) return true; else if (sgn(w)==0) return false;
        if (sgn(x)*sgn(y)<=0&&sgn(z)*sgn(w)<=0) return true; else
return false;
}
inline P work(P a,P b,P c,P d){
        P ans;
        double a1=b.y-a.y,b1=-(b.x-a.x),c1=a.x*b.y-a.y*b.x;
        double a2=d.y-c.y,b2=-(d.x-c.x),c2=c.x*d.y-c.y*d.x;
        ans.x=(c1*b2-c2*b1)/(a1*b2-a2*b1);
ans.y=(a1*c2-a2*c1)/(a1*b2-a2*b1);
        return ans;
}

//Point
const int MAXN=300005,Mo=1000000007;
const double eps=1e-10,INF=1e20,PI=3.14159265358979323846 2;
inline int sgn(double x){ if (fabs(x)<=eps) return 0; return x>0?1:-1; }
struct P{
        double x,y;
        P(double xx=0,double yy=0):x(xx),y(yy){}
        double len(){ return sqrt(x*x+y*y); }
        P operator +(const P &E)const{ return P(x+E.x,y+E.y); }
        P operator -(const P &E)const{ return P(x-E.x,y-E.y); }
        P operator *(const double &f)const{ return P(x*f,y*f); }
        P operator /(const double &f)const{ return P(x/f,y/f); }
        double operator *(const P &E)const{ return x*E.y-E.x*y; }
        double operator ^(const P &E)const{ return x*E.x+y*E.y; }
}A,B,C,O;
inline double dist(P a,P b){ return
sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y)); }
inline double cross(P a,P b,P o){ return (a-o)*(b-o); }
inline double dot(P a,P b,P o){ return (a-o)^(b-o); }
inline double deg(P p){ double tmp=atan2(p.y,p.x); return
tmp<0?tmp+PI*2:tmp; }
inline P rot(P p,double a){ return
P(cos(a)*p.x-sin(a)*p.y,sin(a)*p.x+cos(a)*p.y); }
struct L{
        double a,b,c;
        L(P A,P B){ a=B.y-A.y; b=A.x-B.x; c=-(A*B); }
}l1,l2;
inline P clammer(L l,L r){
        P ans(-l.c*r.b+l.b*r.c,-l.a*r.c+l.c*r.a);
        return ans/(l.a*r.b-r.a*l.b);
}
```

```
int ConvexHull(P p[],int n,P q[]){
        sort(p+1,p+n+1);
        int i,m=0;
        for (i=1;i<=n;i++) {
                while (m>1&&cross(q[m]-q[m-1],p[i]-q[m-1])<=eps)
m--;
                q[++m]=p[i];
        }
        int k=m;
        for (i=n-1;i>0;i--) {
                while (m>k&&cross(q[m]-q[m-1],p[i]-q[m-1])<=eps)
m--;
                q[++m]=p[i];
        }
        if (n>1) m--;
        return m;
}


//旋转卡壳
double Rotating_calipers(P p[],int n,P q[],int m){
        int v,u,i;
        double ans=INF,tmp;
        for (i=u=1;i<=n;++i) if (p[u].y-eps>p[i].y) u=i;
        for (i=v=1;i<=m;++i) if (q[v].y+eps<q[i].y) v=i;
        for (i=1;i<=n;++i) {
                while
((tmp=cross(p[u+1]-p[u],q[v+1]-p[u])-cross(p[u+1]-p[u],q[v]-p[u]))>ep
s)
                        { v++; if (v==m+1) v=1; }
                ans=min(ans,work(p[u],p[u+1],q[v]));
                if (fabs(tmp)<=eps) {
                        ans=min(ans,work(p[u],p[u+1],q[v+1]));
                        ans=min(ans,work(q[v],q[v+1],p[u]));
                        ans=min(ans,work(q[v],q[v+1],p[u+1]));
                }
                u++; if (u==n+1) u=1;
        }
        return ans;
}


//三维凸包
#include<stdio.h>
#include<string.h>
#include<math.h>
#include<algorithm>
#include <iostream>
using namespace std;
#define PR 1e-9
#define N 1100
struct P{
        double x,y,z;
        P(double _x=0,double _y=0,double _z=0):x(_x),y(_y),z(_z){}
        P operator-(const P p) {return P(x-p.x,y-p.y,z-p.z);}
        P operator*(const P p) {return
P(y*p.z-z*p.y,z*p.x-x*p.z,x*p.y-y*p.x);}//叉积
        double operator^(const P p) {return x*p.x+y*p.y+z*p.z;}//点
积
};
P dd;
struct fac{
        int a,b,c;//凸包一个面上的三个点的编号
        bool ok;//该面是否是最终凸包中的面
};
P xmult(P u,P v){
        return P(u.y*v.z-v.y*u.z,u.z*v.x-u.x*v.z,u.x*v.y-u.y*v.x);
}
double dmult(P u,P v){
        return u.x*v.x+u.y*v.y+u.z*v.z;
}
P subt(P u,P v){
        return P(u.x-v.x,u.y-v.y,u.z-v.z);
}
double vlen(P u){
        return sqrt(u.x*u.x+u.y*u.y+u.z*u.z);
}
P pvec(P a,P b,P c){
        return xmult(subt(a,b),subt(b,c));
}
double Dis(P a,P b,P c,P d){
        return fabs(dmult(pvec(a,b,c),subt(d,a)))/vlen(pvec(a,b,c));
}
struct T3dhull{
        int n;//初始点数
        P ply[N];//初始点
        int trianglecnt;//凸包上三角形数
        fac tri[N];//凸包三角形
        int vis[N][N];//点 i 到点 j 是属于哪个面
        double dist(P a){return sqrt(a.x*a.x+a.y*a.y+a.z*a.z);}//两
点长度
        double area(P a,P b,P c){return dist((b-a)*(c-a));}//三角形
面积*2
        double volume(P a,P b,P c,P d){return (b-a)*(c-a)^(d-a);}//
四面体有向体积*6
        double ptoplane(P &p,fac &f){//正：点在面同向
                 P
m=ply[f.b]-ply[f.a],n=ply[f.c]-ply[f.a],t=p-ply[f.a];
                 return (m*n)^t;
        }
        void deal(int p,int a,int b){
                int f=vis[a][b];//与当前面(cnt)共边(ab)的那个面
                fac add;
                if(tri[f].ok){
                        if((ptoplane(ply[p],tri[f]))>PR)
dfs(p,f);//如果 p 点能看到该面 f，则继续深度探索 f 的 3 条边，以便更新新
的凸包面
                        else//否则因为 p 点只看到 cnt 面，看不到 f
面，则 p 点和 a、b 点组成一个三角形。
                        {

add.a=b,add.b=a,add.c=p,add.ok=1;

vis[p][b]=vis[a][p]=vis[b][a]=trianglecnt;
                                 tri[trianglecnt++]=add;
                        }
                }
        }
        void dfs(int p,int cnt){//维护凸包，如果点 p 在凸包外更新凸包
                tri[cnt].ok=0;//当前面需要删除，因为它在更大的凸包
里面
//下面把边反过来(先 b,后 a)，以便在 deal()中判断与当前面(cnt)共边(ab)的
那个面。即判断与当头面(cnt)相邻的 3 个面(它们与当前面的共边是反向的，如
下图中(1)的法线朝外(即逆时针)的面 130 和 312,它们共边 13，但一个方向是
13,另一个方向是 31)
                deal(p,tri[cnt].b,tri[cnt].a);
                deal(p,tri[cnt].c,tri[cnt].b);
                deal(p,tri[cnt].a,tri[cnt].c);
        }
        bool same(int s,int e){//判断两个面是否为同一面
                P a=ply[tri[s].a],b=ply[tri[s].b],c=ply[tri[s].c];
                return fabs(volume(a,b,c,ply[tri[e].a]))<PR
                        &&fabs(volume(a,b,c,ply[tri[e].b]))<PR
                        &&fabs(volume(a,b,c,ply[tri[e].c]))<PR;
        }
        void construct(){//构建凸包
                int i,j;
                trianglecnt=0;
                if(n<4) return ;
                bool tmp=true;
                for(i=1;i<n;i++){//前两点不共点
                        if((dist(ply[0]-ply[i]))>PR)
                        {
                                swap(ply[1],ply[i]); tmp=false;
break;
                        }
        }
        if(tmp) return;
        tmp=true;
        for(i=2;i<n;i++){//前三点不共线

    if((dist((ply[0]-ply[1])*(ply[1]-ply[i])))>PR){
                swap(ply[2],ply[i]); tmp=false;
break;
            }
        }
        if(tmp) return ;
        tmp=true;
        for(i=3;i<n;i++){//前四点不共面

    if(fabs((ply[0]-ply[1])*(ply[1]-ply[2])^(ply[0]-ply[i]))>PR)
{
                swap(ply[3],ply[i]); tmp=false;
break;
            }
        }
        if(tmp) return ;
        fac add;
        for(i=0;i<4;i++){//构建初始四面体(4 个点为
ply[0],ply[1],ply[2],ply[3])

    add.a=(i+1)%4,add.b=(i+2)%4,add.c=(i+3)%4,add.ok=1;
            if((ptoplane(ply[i],add))>0)
swap(add.b,add.c);//保证逆时针，即法向量朝外，这样新点才可看到。

    vis[add.a][add.b]=vis[add.b][add.c]=vis[add.c][add.a]=trian
glecnt;//逆向的有向边保存
            tri[trianglecnt++]=add;
        }
        for(i=4;i<n;i++){//构建更新凸包
            for(j=0;j<trianglecnt;j++){//对每个点判
断是否在当前 3 维凸包内或外(i 表示当前点,j 表示当前面)

    if(tri[j].ok&&(ptoplane(ply[i],tri[j]))>PR){//对当前凸包面
进行判断，看是否点能否看到这个面
                    dfs(i,j); break;//点
能看到当前面，更新凸包的面(递归，可能不止更新一个面)。当前点更新完成后
break 跳出循环
                }
            }
        }
        int cnt=trianglecnt;//这些面中有一些 tri[i].ok=0,
它们属于开始建立但后来因为在更大凸包内故需删除的,所以下面几行代码的作
用是只保存最外层的凸包
        trianglecnt=0;
        for(i=0;i<cnt;i++)
        if(tri[i].ok) tri[trianglecnt++]=tri[i];
    }
    double res(){
        double _min=1e300;
        for(int i=0;i<trianglecnt;i++){
            double
now=Dis(ply[tri[i].a],ply[tri[i].b],ply[tri[i].c],dd);
            if(_min>now) _min=now;
        }
        return _min;
    }
}hull;
int main(){
    while(scanf("%d",&hull.n)!=EOF){
        if(hull.n==0) break;
        int i,j,q;
        for(i=0;i<hull.n;i++)
scanf("%lf%lf%lf",&hull.ply[i].x,&hull.ply[i].y,&hull.ply[i].z);
        hull.construct();
        scanf("%d",&q);
        for(j=0;j<q;j++) {
            scanf("%lf%lf%lf",&dd.x,&dd.y,&dd.z);
            printf("%.4lf\n",hull.res());
        }
    }
    return 0;
}
//hdu4266


//半平面交
#include <cmath>
#include <cstdio>
#include <algorithm>
using namespace std;
const int maxn=1505;
const double eps=1e-8;
int n,pn, dq[maxn], top, bot;//数组模拟双端队列
struct Point{ double x,y;}p[maxn];
struct Line{
    Point a,b;
    double angle;//极角
    Line& operator =(Line l){
        a.x=l.a.x,a.y=l.a.y;
        b.x=l.b.x,b.y=l.b.y;
        angle=l.angle;return *this;
    }
}l[maxn];
int dblcmp(double k){//精度函数
    if (fabs(k)<eps) return 0;
    return k>0?1:-1;
}
double multi(Point p0,Point p1,Point p2){//叉积
    return (p1.x-p0.x)*(p2.y-p0.y)-(p1.y-p0.y)*(p2.x-p0.x);
}
bool cmp(const Line& l1,const Line& l2){
    int d=dblcmp(l1.angle-l2.angle);
    if (!d) return dblcmp(multi(l1.a,l2.a,l2.b))<0;
    //大于 0 取半平面的左半，小于 0 取右半
    return d<0;
}
void addLine(Line& l,double x1,double y1,double x2,double y2){
    l.a.x=x1;l.a.y=y1;
    l.b.x=x2;l.b.y=y2;
    l.angle=atan2(y2-y1,x2-x1);
}
void getIntersect(Line l1,Line l2,Point& p){
    double A1=l1.b.y-l1.a.y;
    double B1=l1.a.x-l1.b.x;
    double C1=(l1.b.x-l1.a.x)*l1.a.y-(l1.b.y-l1.a.y)*l1.a.x;
    double A2=l2.b.y-l2.a.y;
    double B2=l2.a.x-l2.b.x;
    double C2=(l2.b.x-l2.a.x)*l2.a.y-(l2.b.y-l2.a.y)*l2.a.x;
    p.x=(C2*B1-C1*B2)/(A1*B2-A2*B1);
    p.y=(C1*A2-C2*A1)/(A1*B2-A2*B1);
}
bool judge(Line l0,Line l1,Line l2){
    Point p;getIntersect(l1,l2,p);
    return dblcmp(multi(p,l0.a,l0.b))>0;
    //与上面的注释处的大于小于符号相反，大于 0，是 p 在向量
l0.a->l0.b 的左边，小于 0 是在右边，当 p 不在半平面 l0 内时，返回 true
}
void HalfPlaneIntersect(){
    int i,j;
    sort(l,l+n,cmp);//极角排序
    for (i=0,j=0;i<n;i++)
    if (dblcmp(l[i].angle-l[j].angle)>0) l[++j]=l[i];//排除极角
相同（从了 l[1]开始比较）
    n=j+1;//个数
    dq[0]=0; dq[1]=1;
    top=1; bot=0;
    for (i=2;i<n;i++){
        while
(top>bot&&judge(l[i],l[dq[top]],l[dq[top-1]])) top--;
        while
(top>bot&&judge(l[i],l[dq[bot]],l[dq[bot+1]])) bot++;
```

```
                dq[++top]=i;
        }
        while (top>bot&&judge(l[dq[bot]],l[dq[top]],l[dq[top-1]]))
top--;
        while (top>bot&&judge(l[dq[top]],l[dq[bot]],l[dq[bot+1]]))
bot++;
        dq[++top]=dq[bot];
        for (pn=0,i=bot;i<top;i++,pn++)
getIntersect(l[dq[i+1]],l[dq[i]],p[pn]);//更新重复利用 p 数组
}
double getArea(){
        if (pn<3) return 0;
        double area=0;
        for (int i=1;i<pn-1;i++) area+=multi(p[0],p[i],p[i+1]);//利
用 p 数组求面积
        return fabs(area)/2;
}
int main(){
        int t,i;
        scanf ("%d",&t);
        while (t--){
                scanf ("%d",&n);
                for (i=0;i<n;i++) scanf ("%lf%lf",&p[i].x,&p[i].y);
                for (i=0;i<n-1;i++)
addLine(l[i],p[i].x,p[i].y,p[i+1].x,p[i+1].y);
                addLine(l[i],p[i].x,p[i].y,p[0].x,p[0].y);
                HalfPlaneIntersect();
                printf ("%.2lf\n",getArea());
        }
        return 0;
}
/*
PKU 1279 题为顺时针方向
*/


---------------数论------------------
//exGCD
int extend_gcd(int a,int b,int &x,int &y) {
        if (b==0) { x=1; y=0; return a; }
        else {
                int tmp=extend_gcd(b,a%b,y,x);
                y-=x*(a/b); return tmp;
        }
}
/*
AX+BY=GCD(A,B)
B(Y+A/B*X)+A%B*X=GCD(B,A%B)
*/


//中国剩余定理
int CRT(int a[],int m[],int n) {
        int i,M=1,ans=0;
        for (i=1;i<=n;i++) M*=m[i];
        for (i=1;i<=n;i++) {
                int x,y,Mi=M/m[i];
                extend_Euclid(Mi,m[i],x,y);
                ans=(ans+Mi*x*a[i])%M;
        }
        if (ans<0) ans+=M;
        return ans;
}
/*
Problem:
x=ai(%mi)[0<=i<n]
m0,m1,m2,...,mn-1 两两互质
已知 mi,ai,求 x
Solution:
Mi=pi(mj) [i!=j]
gcd(Mi,mi)=1 --> Mipi+miqi=1
Let ei=Mipi ,then ei={0(%mj)[i!=j],1(%mj)[i==j]}
ans0=(e0a0+e1a1+e2a2+...+en-1an-1)%pi(mi)
```

```
*/


//原根
LL pow_mod(LL a,LL b,LL c) {
        LL ans=1;
        while (b) {
                if (b&1) ans=ans*a%c;
                a=a*a%c; b>>=1;
        }
        return ans;
}
vector<LL> a;
bool g_test(LL g,LL p) {
        for (LL i=0;i<a.size();++i)
        if (pow_mod(g,(p-1)/a[i],p)==1) return 0;
        return 1;
}
LL primitive_root(LL p) {
        LL tmp=p-1;
        for (LL i=2;i*i<=tmp;++i)
        if (!(tmp%i)) {
                a.push_back(i);
                while (!(tmp%i)) tmp/=i;
        }
        if (tmp>1) a.push_back(tmp);
        for (LL g=1;;g++) if (g_test(g,p)) return g;
}


//Miller_Rabin
LL add(LL a,LL b,LL c) {
        LL ans=0;
        while (b) {
                if (b&1) {
                        ans+=a;
                        if (ans>=c) ans-=c;
                }
                a<<=1; b>>=1;
                if (a>=c) a-=c;
        }
        return ans;
}
LL power(LL a,LL b,LL c) {
        LL ans=1;
        while (b) {
                if (b&1) ans=add(ans,a,c);
                a=add(a,a,c); b>>=1;
        }
        return ans;
}
bool Miller_Rabin_test(LL a,LL b,LL n) {
        if (n==2) return true;
        else if (n==a||!(n&1)) return false;
        while (!(b&1)) b>>=1;
        LL tmp=power(a,b,n);
        while (b!=n-1&&tmp!=n-1&&tmp!=1) { tmp=add(tmp,tmp,n);
b<<=1; }
        return (tmp==n-1)||((b&1)&&(tmp==1));
}
bool isprime(LL n) {
        int a[]={2,3,7,61,24251}; //10^16
        if (n==46856248255981LL) return false;
        for (int i=0;i<5;++i) {
                if (n==a[i]) return true;
                if (!Miller_Rabin_test(a[i],n-1,n)) return false;
        }
        return true;
}


//exBSGS
#include <cmath>
```

```
#include <cstdio>
#include <cstring>
#include <iostream>
#include <algorithm>
using namespace std;
typedef long long LL;
const int Mo 131071;
LL top;
bool flag[Mo*2];
struct HashNode{ LL data,id,next; }hash[Mo*2];
void Insert(LL a,LL b) {
    LL k=b&Mo;
    if (flag[k]==false) {
        flag[k]=true; hash[k].next=-1; hash[k].id=a;
hash[k].data=b;
        return;
    }
    while (hash[k].next!=-1) {
        if (hash[k].data==b) return;
        k=hash[k].next;
    }
    if (hash[k].data==b) return;
    hash[k].next=++top; hash[top].next=-1; hash[top].id=a;
hash[top].data=b;
}
LL Find(LL b) {
    LL k=b&Mo;
    if(flag[k]==false) return -1;
    while (k!=-1) {
        if (hash[k].data==b) return hash[k].id;
        k=hash[k].next;
    }
    return -1;
}
LL gcd(LL a,LL b) { return b?gcd(b,a%b):a; }
LL ext_gcd (LL a, LL b, LL& x, LL& y ) {
    LL t,ret;
    if (b==0) { x=1,y=0; return a; }
    ret=ext_gcd(b,a%b,x,y); t=x; x=y; y=t-a/b*y;
    return ret;
}
LL mod_exp(LL a,LL b,LL n) {
    LL ret=1; a=a%n;
    while (b>=1) {
        if (b&1) ret=ret*a%n;
        a=a*a%n; b>>=1;
    }
    return ret;
}
LL BabyStep_GiantStep(LL A,LL B,LL C) {
    top=Mo; B%=C;
    LL i,tmp=1;
    for (i=0;i<=100;tmp=tmp*A%C,i++) if (tmp==B%C) return i;
    LL D=1,cnt=0;
    while ((tmp=gcd(A,C))!=1) {
        if (B%tmp) return -1;
        C/=tmp; B/=tmp; D=D*A/tmp%C; cnt++;
    }
    LL M=(LL)sqrt(C+0.0);
    for (tmp=1,i=0;i<=M;tmp=tmp*A%C,i++) Insert(i,tmp);
    LL x,y,K=mod_exp(A,M,C);
    for (i=0;i<=M;i++) {
        ext_gcd(D,C,x,y); //D*X=1(mod C)
        tmp=((B*x)%C+C)%C;
        if((y=Find(tmp))!=-1) return i*M+y+cnt;
        D=D*K%C;
    }
    return -1;
}
int main() {
    LL A,B,C;
    scanf("%lld%lld%lld",&A,&C,&B);
    while (!(A==0&&B==0&&C==0)) {
```

```
        memset(flag,0,sizeof(flag));
        LL tmp=BabyStep_GiantStep(A,B,C);
        if (tmp==-1) puts("No Solution\n"); else
printf("%lld\n",tmp);
        scanf("%lld%lld%lld",&A,&C,&B);
    }
    return 0;
}
/*
扩展BSGS
简单的来说，就是加了一个把不互质的数通过去除公因数变为互质的，再进行
BSGS
具体的就是
考虑a与p不互质的情况：1.对于a^x=b mod p，我们可以考虑从x个a中拿
出c个a与b和p消去公因子，直到a和p’互质为止。
2.一旦互质了，那么方程就是v*a^(x-c)=b’ (mod p’)，v是拿出c个a消去
公因子后剩下的东西，b’，p’是消去公因子的b，p。
这个时候还要求v的逆元，方程变为a^(x-c)=b’*v^(-1) (mod p’)。
此时就可以用baby-step-giant-step做了，答案为BSGS的答案+c。
注意:有可能c会大于x,所以必须约去一次就特判一次方程两边(就是v和b’)
是不是相等了。如果两边相等那么直接返回c。
*/


//FWT
const int Mo=1000000007,rev=500000004;
void FWT(int a[],int n) {
    for(int d=1;d<n;d<<=1)
        for(int m=d<<1,i=0;i<n;i+=m)
            for(int j=0;j<d;j++) {
                int x=a[i+j],y=a[i+j+d];

    a[i+j]=(x+y)%mod,a[i+j+d]=(x-y+mod)%mod;

    //xor:a[i+j]=x+y,a[i+j+d]=(x-y+mod)%mod;

    //and:a[i+j]=x+y,a[i+j+d]=a[i+j+d];

    //or:a[i+j]=a[i+j],a[i+j+d]=x+y;
            }
}
void UFWT(int a[],int n) {
    for(int d=1;d<n;d<<=1)
        for(int i=0,m=d<<1;i<n;i+=m)
            for(int j=0;j<d;j++) {
                int x=a[i+j],y=a[i+j+d];

    a[i+j]=1LL*(x+y)*rev%mod,a[i+j+d]=(1LL*(x-y)*rev%mod+mod)%m
od;

    //xor:a[i+j]=(x+y)/2,a[i+j+d]=(x-y)/2;

    //and:a[i+j]=x-y,a[i+j+d]=a[i+j+d];

    //or:a[i+j]=a[i+j],a[i+j+d]=y-x;
            }
}
void solve(int a[],int b[],int n) {
    FWT(a,n); FWT(b,n);
    for(int i=0;i<n;i++) a[i]=1LL*a[i]*b[i]%mod;
    UFWT(a,n);
}
//当指异或非运算、与非运算、或非运算时,我们可以将 直接用异或运算、与运
算、或运算的方法求出来,然后将互反的两位交换即可
```