```
#include<vector>
#include<list>
#include<map>
#include<set>
#include<deque>
#include<queue>
#include<stack>
#include<bitset>
#include<algorithm>
#include<functional>
#include<numeric>
#include<utility>
#include<iostream>
#include<sstream>
#include<iomanip>
#include<cstdio>
#include<cmath>
#include<cstdlib>
#include<cctype>
#include<string>
#include<cstring>
#include<cstdio>
#include<cmath>
#include<cstdlib>
#include<ctime>
#include<climits>
#include<complex>
#define mp make_pair
#define pb push_back
using namespace std;
const double eps=1e-8;
const double pi=acos(-1.0);
const double inf=1e20;
const int maxp=1111;
int dblcmp(double d)
{
    if (fabs(d)<eps)return 0;
    return d>eps?1:-1;
}
inline double sqr(double x){return x*x;}
struct point
{
    double x,y;
    point(){}
    point(double _x,double _y):
    x(_x),y(_y){};
    void input()
    {
        scanf("%lf%lf",&x,&y);
    }
    void output()
    {
        printf("%.2f %.2f\n",x,y);
    }
    bool operator==(point a)const
    {
```

```
        return dblcmp(a.x-x)==0&&dblcmp(a.y-y)==0;
    }
    bool operator<(point a)const
    {
        return dblcmp(a.x-x)==0?dblcmp(y-a.y)<0:x<a.x;
    }
    double len()
    {
        return hypot(x,y);
    }
    double len2()
    {
        return x*x+y*y;
    }
    double distance(point p)
    {
        return hypot(x-p.x,y-p.y);
    }
    point add(point p)
    {
        return point(x+p.x,y+p.y);
    }
    point sub(point p)
    {
        return point(x-p.x,y-p.y);
    }
    point mul(double b)
    {
        return point(x*b,y*b);
    }
    point div(double b)
    {
        return point(x/b,y/b);
    }
    double dot(point p)
    {
        return x*p.x+y*p.y;
    }
    double det(point p)
    {
        return x*p.y-y*p.x;
    }
    double rad(point a,point b)
    {
        point p=*this;
        return fabs(atan2(fabs(a.sub(p).det(b.sub(p))),a.sub(p).dot(b.sub(p))));
    }
    point trunc(double r)
    {
        double l=len();
        if (!dblcmp(l))return *this;
        r/=l;
        return point(x*r,y*r);
    }
    point rotleft()
    {
```

```
        return point(-y,x);
    }
    point rotright()
    {
        return point(y,-x);
    }
    point rotate(point p,double angle)//绕点p逆时针旋转angle角度
    {
        point v=this->sub(p);
        double c=cos(angle),s=sin(angle);
        return point(p.x+v.x*c-v.y*s,p.y+v.x*s+v.y*c);
    }
};
struct line
{
    point a,b;
    line(){}
    line(point _a,point _b)
    {
        a=_a;
        b=_b;
    }
    bool operator==(line v)
    {
        return (a==v.a)&&(b==v.b);
    }
    //倾斜角angle
    line(point p,double angle)
    {
        a=p;
        if (dblcmp(angle-pi/2)==0)
        {
            b=a.add(point(0,1));
        }
        else
        {
            b=a.add(point(1,tan(angle)));
        }
    }
    //ax+by+c=0
    line(double _a,double _b,double _c)
    {
        if (dblcmp(_a)==0)
        {
            a=point(0,-_c/_b);
            b=point(1,-_c/_b);
        }
        else if (dblcmp(_b)==0)
        {
            a=point(-_c/_a,0);
            b=point(-_c/_a,1);
        }
        else
        {
            a=point(0,-_c/_b);
```

```
            b=point(1,(-_c-_a)/_b);
        }
    }
    void input()
    {
        a.input();
        b.input();
    }
    void adjust()
    {
        if (b<a)swap(a,b);
    }
    double length()
    {
        return a.distance(b);
    }
    double angle()//直线倾斜角 0<=angle<180
    {
        double k=atan2(b.y-a.y,b.x-a.x);
        if (dblcmp(k)<0)k+=pi;
        if (dblcmp(k-pi)==0)k-=pi;
        return k;
    }
    //点和线段关系
    //1 在逆时针
    //2 在顺时针
    //3 平行
    int relation(point p)
    {
        int c=dblcmp(p.sub(a).det(b.sub(a)));
        if (c<0)return 1;
        if (c>0)return 2;
        return 3;
    }
    bool pointonseg(point p)
    {
        return
dblcmp(p.sub(a).det(b.sub(a)))==0&&dblcmp(p.sub(a).dot(p.sub(b)))<=0;
    }
    bool parallel(line v)
    {
        return dblcmp(b.sub(a).det(v.b.sub(v.a)))==0;
    }
    //2 规范相交
    //1 非规范相交
    //0 不相交
    int segcrossseg(line v)
    {
        int d1=dblcmp(b.sub(a).det(v.a.sub(a)));
        int d2=dblcmp(b.sub(a).det(v.b.sub(a)));
        int d3=dblcmp(v.b.sub(v.a).det(a.sub(v.a)));
        int d4=dblcmp(v.b.sub(v.a).det(b.sub(v.a)));
        if ((d1^d2)==-2&&(d3^d4)==-2)return 2;
        return (d1==0&&dblcmp(v.a.sub(a).dot(v.a.sub(b)))<=0||
                d2==0&&dblcmp(v.b.sub(a).dot(v.b.sub(b)))<=0||
```

```
                d3==0&&dblcmp(a.sub(v.a).dot(a.sub(v.b)))<=0||
                d4==0&&dblcmp(b.sub(v.a).dot(b.sub(v.b)))<=0);
    }
    int linecrossseg(line v)//*this seg v line
    {
        int d1=dblcmp(b.sub(a).det(v.a.sub(a)));
        int d2=dblcmp(b.sub(a).det(v.b.sub(a)));
        if ((d1^d2)==-2)return 2;
        return (d1==0||d2==0);
    }
    //0 平行
    //1 重合
    //2 相交
    int linecrossline(line v)
    {
        if ((*this).parallel(v))
        {
            return v.relation(a)==3;
        }
        return 2;
    }
    point crosspoint(line v)
    {
        double a1=v.b.sub(v.a).det(a.sub(v.a));
        double a2=v.b.sub(v.a).det(b.sub(v.a));
        return point((a.x*a2-b.x*a1)/(a2-a1),(a.y*a2-b.y*a1)/(a2-a1));
    }
    double dispointtoline(point p)
    {
        return fabs(p.sub(a).det(b.sub(a)))/length();
    }
    double dispointtoseg(point p)
    {
        if (dblcmp(p.sub(b).dot(a.sub(b)))<0||dblcmp(p.sub(a).dot(b.sub(a)))<0)
        {
            return min(p.distance(a),p.distance(b));
        }
        return dispointtoline(p);
    }
    point lineprog(point p)
    {
        return a.add(b.sub(a).mul(b.sub(a).dot(p.sub(a))/b.sub(a).len2()));
    }
    point symmetrypoint(point p)
    {
        point q=lineprog(p);
        return point(2*q.x-p.x,2*q.y-p.y);
    }
};
struct circle
{
    point p;
    double r;
    circle(){}
    circle(point _p,double _r):
```

```
    p(_p),r(_r){};
    circle(double x,double y,double _r):
    p(point(x,y)),r(_r){};
    circle(point a,point b,point c)//三角形的外接圆
    {

p=line(a.add(b).div(2),a.add(b).div(2).add(b.sub(a).rotleft())).crosspoint(line(
c.add(b).div(2),c.add(b).div(2).add(b.sub(c).rotleft())));
        r=p.distance(a);
    }
    circle(point a,point b,point c,bool t)//三角形的内切圆
    {
        line u,v;
        double m=atan2(b.y-a.y,b.x-a.x),n=atan2(c.y-a.y,c.x-a.x);
        u.a=a;
        u.b=u.a.add(point(cos((n+m)/2),sin((n+m)/2)));
        v.a=b;
        m=atan2(a.y-b.y,a.x-b.x),n=atan2(c.y-b.y,c.x-b.x);
        v.b=v.a.add(point(cos((n+m)/2),sin((n+m)/2)));
        p=u.crosspoint(v);
        r=line(a,b).dispointtoseg(p);
    }
    void input()
    {
        p.input();
        scanf("%lf",&r);
    }
    void output()
    {
        printf("%.2lf %.2lf %.2lf\n",p.x,p.y,r);
    }
    bool operator==(circle v)
    {
        return ((p==v.p)&&dblcmp(r-v.r)==0);
    }
    bool operator<(circle v)const
    {
        return ((p<v.p)||(p==v.p)&&dblcmp(r-v.r)<0);
    }
    double area()
    {
        return pi*sqr(r);
    }
    double circumference()
    {
        return 2*pi*r;
    }
    //0 圆外
    //1 圆上
    //2 圆内
    int relation(point b)
    {
        double dst=b.distance(p);
        if (dblcmp(dst-r)<0)return 2;
        if (dblcmp(dst-r)==0)return 1;
```

```
        return 0;
    }
    int relationseg(line v)
    {
        double dst=v.dispointtoseg(p);
        if (dblcmp(dst-r)<0)return 2;
        if (dblcmp(dst-r)==0)return 1;
        return 0;
    }
    int relationline(line v)
    {
        double dst=v.dispointtoline(p);
        if (dblcmp(dst-r)<0)return 2;
        if (dblcmp(dst-r)==0)return 1;
        return 0;
    }
    //过a b两点 半径r的两个圆
    int getcircle(point a,point b,double r,circle&c1,circle&c2)
    {
        circle x(a,r),y(b,r);
        int t=x.pointcrosscircle(y,c1.p,c2.p);
        if (!t)return 0;
        c1.r=c2.r=r;
        return t;
    }
    //与直线u相切 过点q 半径r1的圆
    int getcircle(line u,point q,double r1,circle &c1,circle &c2)
    {
        double dis=u.dispointtoline(q);
        if (dblcmp(dis-r1*2)>0)return 0;
        if (dblcmp(dis)==0)
        {
            c1.p=q.add(u.b.sub(u.a).rotleft().trunc(r1));
            c2.p=q.add(u.b.sub(u.a).rotright().trunc(r1));
            c1.r=c2.r=r1;
            return 2;
        }
        line
u1=line(u.a.add(u.b.sub(u.a).rotleft().trunc(r1)),u.b.add(u.b.sub(u.a).rotleft()
.trunc(r1)));
        line
u2=line(u.a.add(u.b.sub(u.a).rotright().trunc(r1)),u.b.add(u.b.sub(u.a).rotright
().trunc(r1)));
        circle cc=circle(q,r1);
        point p1,p2;
        if (!cc.pointcrossline(u1,p1,p2))cc.pointcrossline(u2,p1,p2);
        c1=circle(p1,r1);
        if (p1==p2)
        {
            c2=c1;return 1;
        }
        c2=circle(p2,r1);
        return 2;
    }
    //同时与直线u,v相切 半径r1的圆
```

```
    int getcircle(line u,line v,double r1,circle &c1,circle &c2,circle
&c3,circle &c4)
    {
        if (u.parallel(v))return 0;
        line
u1=line(u.a.add(u.b.sub(u.a).rotleft().trunc(r1)),u.b.add(u.b.sub(u.a).rotleft()
.trunc(r1)));
        line
u2=line(u.a.add(u.b.sub(u.a).rotright().trunc(r1)),u.b.add(u.b.sub(u.a).rotright
().trunc(r1)));
        line
v1=line(v.a.add(v.b.sub(v.a).rotleft().trunc(r1)),v.b.add(v.b.sub(v.a).rotleft()
.trunc(r1)));
        line
v2=line(v.a.add(v.b.sub(v.a).rotright().trunc(r1)),v.b.add(v.b.sub(v.a).rotright
().trunc(r1)));
        c1.r=c2.r=c3.r=c4.r=r1;
        c1.p=u1.crosspoint(v1);
        c2.p=u1.crosspoint(v2);
        c3.p=u2.crosspoint(v1);
        c4.p=u2.crosspoint(v2);
        return 4;
    }
    //同时与不相交圆cx,cy相切 半径为r1的圆
    int getcircle(circle cx,circle cy,double r1,circle&c1,circle&c2)
    {
        circle x(cx.p,r1+cx.r),y(cy.p,r1+cy.r);
        int t=x.pointcrosscircle(y,c1.p,c2.p);
        if (!t)return 0;
        c1.r=c2.r=r1;
        return t;
    }
    int pointcrossline(line v,point &p1,point &p2)//求与线段交要先判断relationseg
    {
        if (!(*this).relationline(v))return 0;
        point a=v.lineprog(p);
        double d=v.dispointtoline(p);
        d=sqrt(r*r-d*d);
        if (dblcmp(d)==0)
        {
            p1=a;
            p2=a;
            return 1;
        }
        p1=a.sub(v.b.sub(v.a).trunc(d));
        p2=a.add(v.b.sub(v.a).trunc(d));
        return 2;
    }
    //5 相离
    //4 外切
    //3 相交
    //2 内切
    //1 内含
    int relationcircle(circle v)
    {
```

```
    double d=p.distance(v.p);
    if (dblcmp(d-r-v.r)>0)return 5;
    if (dblcmp(d-r-v.r)==0)return 4;
    double l=fabs(r-v.r);
    if (dblcmp(d-r-v.r)<0&&dblcmp(d-l)>0)return 3;
    if (dblcmp(d-l)==0)return 2;
    if (dblcmp(d-l)<0)return 1;
}
int pointcrosscircle(circle v,point &p1,point &p2)
{
    int rel=relationcircle(v);
    if (rel==1||rel==5)return 0;
    double d=p.distance(v.p);
    double l=(d+(sqr(r)-sqr(v.r))/d)/2;
    double h=sqrt(sqr(r)-sqr(l));
    p1=p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).rotleft().trunc(h)));
    p2=p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).rotright().trunc(h)));
    if (rel==2||rel==4)
    {
        return 1;
    }
    return 2;
}
//过一点做圆的切线（先判断点和圆关系）
int tangentline(point q,line &u,line &v)
{
    int x=relation(q);
    if (x==2)return 0;
    if (x==1)
    {
        u=line(q,q.add(q.sub(p).rotleft()));
        v=u;
        return 1;
    }
    double d=p.distance(q);
    double l=sqr(r)/d;
    double h=sqrt(sqr(r)-sqr(l));
    u=line(q,p.add(q.sub(p).trunc(l).add(q.sub(p).rotleft().trunc(h))));
    v=line(q,p.add(q.sub(p).trunc(l).add(q.sub(p).rotright().trunc(h))));
    return 2;
}
double areacircle(circle v)
{
    int rel=relationcircle(v);
    if (rel>=4)return 0.0;
    if (rel<=2)return min(area(),v.area());
    double d=p.distance(v.p);
    double hf=(r+v.r+d)/2.0;
    double ss=2*sqrt(hf*(hf-r)*(hf-v.r)*(hf-d));
    double a1=acos((r*r+d*d-v.r*v.r)/(2.0*r*d));
    a1=a1*r*r;
    double a2=acos((v.r*v.r+d*d-r*r)/(2.0*v.r*d));
    a2=a2*v.r*v.r;
    return a1+a2-ss;
}
double areatriangle(point a,point b)
```

```
    {
        if (dblcmp(p.sub(a).det(p.sub(b))==0))return 0.0;
        point q[5];
        int len=0;
        q[len++]=a;
        line l(a,b);
        point p1,p2;
        if (pointcrossline(l,q[1],q[2])==2)
        {
            if (dblcmp(a.sub(q[1]).dot(b.sub(q[1])))<0)q[len++]=q[1];
            if (dblcmp(a.sub(q[2]).dot(b.sub(q[2])))<0)q[len++]=q[2];
        }
        q[len++]=b;
        if
(len==4&&(dblcmp(q[0].sub(q[1]).dot(q[2].sub(q[1])))>0))swap(q[1],q[2]);
        double res=0;
        int i;
        for (i=0;i<len-1;i++)
        {
            if (relation(q[i])==0||relation(q[i+1])==0)
            {
                double arg=p.rad(q[i],q[i+1]);
                res+=r*r*arg/2.0;
            }
            else
            {
                res+=fabs(q[i].sub(p).det(q[i+1].sub(p))/2.0);
            }
        }
        return res;
    }
};
struct polygon
{
    int n;
    point p[maxp];
    line l[maxp];
    void input()
    {
        n=4;
        for (int i=0;i<n;i++)
        {
            p[i].input();
        }
    }
    void add(point q)
    {
        p[n++]=q;
    }
    void getline()
    {
        for (int i=0;i<n;i++)
        {
            l[i]=line(p[i],p[(i+1)%n]);
        }
    }
```

```
    struct cmp
    {
        point p;
        cmp(const point &p0){p=p0;}
        bool operator()(const point &aa,const point &bb)
        {
            point a=aa,b=bb;
            int d=dblcmp(a.sub(p).det(b.sub(p)));
            if (d==0)
            {
                return dblcmp(a.distance(p)-b.distance(p))<0;
            }
            return d>0;
        }
    };
    void norm()
    {
        point mi=p[0];
        for (int i=1;i<n;i++)mi=min(mi,p[i]);
        sort(p,p+n,cmp(mi));
    }
    void getconvex(polygon &convex)
    {
        int i,j,k;
        sort(p,p+n);
        convex.n=n;
        for (i=0;i<min(n,2);i++)
        {
            convex.p[i]=p[i];
        }
        if (n<=2)return;
        int &top=convex.n;
        top=1;
        for (i=2;i<n;i++)
        {
            while
(top&&convex.p[top].sub(p[i]).det(convex.p[top-1].sub(p[i]))<=0)
                top--;
            convex.p[++top]=p[i];
        }
        int temp=top;
        convex.p[++top]=p[n-2];
        for (i=n-3;i>=0;i--)
        {
            while (top!
=temp&&convex.p[top].sub(p[i]).det(convex.p[top-1].sub(p[i]))<=0)
                top--;
            convex.p[++top]=p[i];
        }
    }
    bool isconvex()
    {
        bool s[3];
        memset(s,0,sizeof(s));
        int i,j,k;
        for (i=0;i<n;i++)
```

```
        {
            j=(i+1)%n;
            k=(j+1)%n;
            s[dblcmp(p[j].sub(p[i]).det(p[k].sub(p[i])))+1]=1;
            if (s[0]&&s[2])return 0;
        }
        return 1;
    }
    //3 点上
    //2 边上
    //1 内部
    //0 外部
    int relationpoint(point q)
    {
        int i,j;
        for (i=0;i<n;i++)
        {
            if (p[i]==q)return 3;
        }
        getline();
        for (i=0;i<n;i++)
        {
            if (l[i].pointonseg(q))return 2;
        }
        int cnt=0;
        for (i=0;i<n;i++)
        {
            j=(i+1)%n;
            int k=dblcmp(q.sub(p[j]).det(p[i].sub(p[j])));
            int u=dblcmp(p[i].y-q.y);
            int v=dblcmp(p[j].y-q.y);
            if (k>0&&u<0&&v>=0)cnt++;
            if (k<0&&v<0&&u>=0)cnt--;
        }
        return cnt!=0;
    }
    //1 在多边形内长度为正
    //2 相交或与边平行
    //0 无任何交点
    int relationline(line u)
    {
        int i,j,k=0;
        getline();
        for (i=0;i<n;i++)
        {
            if (l[i].segcrossseg(u)==2)return 1;
            if (l[i].segcrossseg(u)==1)k=1;
        }
        if (!k)return 0;
        vector<point>vp;
        for (i=0;i<n;i++)
        {
            if (l[i].segcrossseg(u))
            {
                if (l[i].parallel(u))
```

```
            {
                vp.pb(u.a);
                vp.pb(u.b);
                vp.pb(l[i].a);
                vp.pb(l[i].b);
                continue;
            }
            vp.pb(l[i].crosspoint(u));
        }
    }
    sort(vp.begin(),vp.end());
    int sz=vp.size();
    for (i=0;i<sz-1;i++)
    {
        point mid=vp[i].add(vp[i+1]).div(2);
        if (relationpoint(mid)==1)return 1;
    }
    return 2;
}
//直线u切割凸多边形左侧
//注意直线方向
void convexcut(line u,polygon &po)
{
    int i,j,k;
    int &top=po.n;
    top=0;
    for (i=0;i<n;i++)
    {
        int d1=dblcmp(p[i].sub(u.a).det(u.b.sub(u.a)));
        int d2=dblcmp(p[(i+1)%n].sub(u.a).det(u.b.sub(u.a)));
        if (d1>=0)po.p[top++]=p[i];
        if (d1*d2<0)po.p[top++]=u.crosspoint(line(p[i],p[(i+1)%n]));
    }
}
double getcircumference()
{
    double sum=0;
    int i;
    for (i=0;i<n;i++)
    {
        sum+=p[i].distance(p[(i+1)%n]);
    }
    return sum;
}
double getarea()
{
    double sum=0;
    int i;
    for (i=0;i<n;i++)
    {
        sum+=p[i].det(p[(i+1)%n]);
    }
    return fabs(sum)/2;
}
bool getdir()//1代表逆时针 0代表顺时针
```

```
{
    double sum=0;
    int i;
    for (i=0;i<n;i++)
    {
        sum+=p[i].det(p[(i+1)%n]);
    }
    if (dblcmp(sum)>0)return 1;
    return 0;
}
point getbarycentre()
{
    point ret(0,0);
    double area=0;
    int i;
    for (i=1;i<n-1;i++)
    {
        double tmp=p[i].sub(p[0]).det(p[i+1].sub(p[0]));
        if (dblcmp(tmp)==0)continue;
        area+=tmp;
        ret.x+=(p[0].x+p[i].x+p[i+1].x)/3*tmp;
        ret.y+=(p[0].y+p[i].y+p[i+1].y)/3*tmp;
    }
    if (dblcmp(area))ret=ret.div(area);
    return ret;
}
double areaintersection(polygon po)
{
}
double areaunion(polygon po)
{
    return getarea()+po.getarea()-areaintersection(po);
}
double areacircle(circle c)
{
    int i,j,k,l,m;
    double ans=0;
    for (i=0;i<n;i++)
    {
        int j=(i+1)%n;
        if (dblcmp(p[j].sub(c.p).det(p[i].sub(c.p)))>=0)
        {
            ans+=c.areatriangle(p[i],p[j]);
        }
        else
        {
            ans-=c.areatriangle(p[i],p[j]);
        }
    }
    return fabs(ans);
}
//多边形和圆关系
//0 一部分在圆外
//1 与圆某条边相切
//2 完全在圆内
```

```
int relationcircle(circle c)
{
    getline();
    int i,x=2;
    if (relationpoint(c.p)!=1)return 0;
    for (i=0;i<n;i++)
    {
        if (c.relationseg(l[i])==2)return 0;
        if (c.relationseg(l[i])==1)x=1;
    }
    return x;
}
void find(int st,point tri[],circle &c)
{
    if (!st)
    {
        c=circle(point(0,0),-2);
    }
    if (st==1)
    {
        c=circle(tri[0],0);
    }
    if (st==2)
    {
        c=circle(tri[0].add(tri[1]).div(2),tri[0].distance(tri[1])/2.0);
    }
    if (st==3)
    {
        c=circle(tri[0],tri[1],tri[2]);
    }
}
void solve(int cur,int st,point tri[],circle &c)
{
    find(st,tri,c);
    if (st==3)return;
    int i;
    for (i=0;i<cur;i++)
    {
        if (dblcmp(p[i].distance(c.p)-c.r)>0)
        {
            tri[st]=p[i];
            solve(i,st+1,tri,c);
        }
    }
}
circle mincircle()//点集最小圆覆盖
{
    random_shuffle(p,p+n);
    point tri[4];
    circle c;
    solve(n,0,tri,c);
    return c;
}
int circlecover(double r)//单位圆覆盖
{
```

```
        int ans=0,i,j;
        vector<pair<double,int> >v;
        for (i=0;i<n;i++)
        {
            v.clear();
            for (j=0;j<n;j++)if (i!=j)
            {
                point q=p[i].sub(p[j]);
                double d=q.len();
                if (dblcmp(d-2*r)<=0)
                {
                    double arg=atan2(q.y,q.x);
                    if (dblcmp(arg)<0)arg+=2*pi;
                    double t=acos(d/(2*r));
                    v.push_back(make_pair(arg-t+2*pi,-1));
                    v.push_back(make_pair(arg+t+2*pi,1));
                }
            }
            sort(v.begin(),v.end());
            int cur=0;
            for (j=0;j<v.size();j++)
            {
                if (v[j].second==-1)++cur;
                else --cur;
                ans=max(ans,cur);
            }
        }
        return ans+1;
    }
    int pointinpolygon(point q)//点在凸多边形内部的判定
    {
        if (getdir())reverse(p,p+n);
        if (dblcmp(q.sub(p[0]).det(p[n-1].sub(p[0])))==0)
        {
            if (line(p[n-1],p[0]).pointonseg(q))return n-1;
            return -1;
        }
        int low=1,high=n-2,mid;
        while (low<=high)
        {
            mid=(low+high)>>1;
            if
(dblcmp(q.sub(p[0]).det(p[mid].sub(p[0])))>=0&&dblcmp(q.sub(p[0]).det(p[mid
+1].sub(p[0])))<0)
            {
                polygon c;
                c.p[0]=p[mid];
                c.p[1]=p[mid+1];
                c.p[2]=p[0];
                c.n=3;
                if (c.relationpoint(q))return mid;
                return -1;
            }
            if (dblcmp(q.sub(p[0]).det(p[mid].sub(p[0])))>0)
            {
                low=mid+1;
```

```
            }
            else
            {
                high=mid-1;
            }
        }
        return -1;
    }
};
struct polygons
{
    vector<polygon>p;
    polygons()
    {
        p.clear();
    }
    void clear()
    {
        p.clear();
    }
    void push(polygon q)
    {
        if (dblcmp(q.getarea()))p.pb(q);
    }
    vector<pair<double,int> >e;
    void ins(point s,point t,point X,int i)
    {
        double r=fabs(t.x-s.x)>eps?(X.x-s.x)/(t.x-s.x):(X.y-s.y)/(t.y-s.y);
        r=min(r,1.0);r=max(r,0.0);
        e.pb(mp(r,i));
    }
    double polyareaunion()
    {
        double ans=0.0;
        int c0,c1,c2,i,j,k,w;
        for (i=0;i<p.size();i++)
        {
            if (p[i].getdir()==0)reverse(p[i].p,p[i].p+p[i].n);
        }
        for (i=0;i<p.size();i++)
        {
            for (k=0;k<p[i].n;k++)
            {
                point &s=p[i].p[k],&t=p[i].p[(k+1)%p[i].n];
                if (!dblcmp(s.det(t)))continue;
                e.clear();
                e.pb(mp(0.0,1));
                e.pb(mp(1.0,-1));
                for (j=0;j<p.size();j++)if (i!=j)
                {
                    for (w=0;w<p[j].n;w++)
                    {
                        point a=p[j].p[w],b=p[j].p[(w
+1)%p[j].n],c=p[j].p[(w-1+p[j].n)%p[j].n];
                        c0=dblcmp(t.sub(s).det(c.sub(s)));
                        c1=dblcmp(t.sub(s).det(a.sub(s)));
```

```
                        c2=dblcmp(t.sub(s).det(b.sub(s)));
                        if (c1*c2<0)ins(s,t,line(s,t).crosspoint(line(a,b)),-
c2);
                        else if (!c1&&c0*c2<0)ins(s,t,a,-c2);
                        else if (!c1&&!c2)
                        {
                            int c3=dblcmp(t.sub(s).det(p[j].p[(w
+2)%p[j].n].sub(s)));
                            int dp=dblcmp(t.sub(s).dot(b.sub(a)));
                            if (dp&&c0)ins(s,t,a,dp>0?c0*((j>i)^(c0<0)):-
(c0<0));
                            if (dp&&c3)ins(s,t,b,dp>0?-c3*((j>i)^(c3<0)):c3<0);
                        }
                    }
                }
                sort(e.begin(),e.end());
                int ct=0;
                double tot=0.0,last;
                for (j=0;j<e.size();j++)
                {
                    if (ct==2)tot+=e[j].first-last;
                    ct+=e[j].second;
                    last=e[j].first;
                }
                ans+=s.det(t)*tot;
            }
        }
        return fabs(ans)*0.5;
    }
};
const int maxn=500;
struct circles
{
    circle c[maxn];
    double ans[maxn];//ans[i]表示被覆盖了i次的面积
    double pre[maxn];
    int n;
    circles(){}
    void add(circle cc)
    {
        c[n++]=cc;
    }
    bool inner(circle x,circle y)
    {
        if (x.relationcircle(y)!=1)return 0;
        return dblcmp(x.r-y.r)<=0?1:0;
    }
    void init_or()//圆的面积并去掉内含的圆
    {
        int i,j,k=0;
        bool mark[maxn]={0};
        for (i=0;i<n;i++)
        {
            for (j=0;j<n;j++)if (i!=j&&!mark[j])
            {
```

```
                if ((c[i]==c[j])||inner(c[i],c[j]))break;
            }
            if (j<n)mark[i]=1;
        }
        for (i=0;i<n;i++)if (!mark[i])c[k++]=c[i];
        n=k;
    }
    void init_and()//圆的面积交去掉内含的圆
    {
        int i,j,k=0;
        bool mark[maxn]={0};
        for (i=0;i<n;i++)
        {
            for (j=0;j<n;j++)if (i!=j&&!mark[j])
            {
                if ((c[i]==c[j])||inner(c[j],c[i]))break;
            }
            if (j<n)mark[i]=1;
        }
        for (i=0;i<n;i++)if (!mark[i])c[k++]=c[i];
        n=k;
    }
    double areaarc(double th,double r)
    {
        return 0.5*sqr(r)*(th-sin(th));
    }
    void getarea()
    {
        int i,j,k;
        memset(ans,0,sizeof(ans));
        vector<pair<double,int> >v;
        for (i=0;i<n;i++)
        {
            v.clear();
            v.push_back(make_pair(-pi,1));
            v.push_back(make_pair(pi,-1));
            for (j=0;j<n;j++)if (i!=j)
            {
                point q=c[j].p.sub(c[i].p);
                double ab=q.len(),ac=c[i].r,bc=c[j].r;
                if (dblcmp(ab+ac-bc)<=0)
                {
                    v.push_back(make_pair(-pi,1));
                    v.push_back(make_pair(pi,-1));
                    continue;
                }
                if (dblcmp(ab+bc-ac)<=0)continue;
                if (dblcmp(ab-ac-bc)>0) continue;
                double th=atan2(q.y,q.x),fai=acos((ac*ac+ab*ab-bc*bc)/
(2.0*ac*ab));
                double a0=th-fai;
                if (dblcmp(a0+pi)<0)a0+=2*pi;
                double a1=th+fai;
                if (dblcmp(a1-pi)>0)a1-=2*pi;
                if (dblcmp(a0-a1)>0)
                {
```

```
                    v.push_back(make_pair(a0,1));
                    v.push_back(make_pair(pi,-1));
                    v.push_back(make_pair(-pi,1));
                    v.push_back(make_pair(a1,-1));
                }
                else
                {
                    v.push_back(make_pair(a0,1));
                    v.push_back(make_pair(a1,-1));
                }
            }
            sort(v.begin(),v.end());
            int cur=0;
            for (j=0;j<v.size();j++)
            {
                if (cur&&dblcmp(v[j].first-pre[cur]))
                {
                    ans[cur]+=areaarc(v[j].first-pre[cur],c[i].r);
                    ans[cur]+=0.5*point(c[i].p.x+c[i].r*cos(pre[cur]),c[i].p.y
+c[i].r*sin(pre[cur])).det(point(c[i].p.x+c[i].r*cos(v[j].first),c[i].p.y
+c[i].r*sin(v[j].first)));
                }
                cur+=v[j].second;
                pre[cur]=v[j].first;
            }
        }
        for (i=1;i<=n;i++)
        {
            ans[i]-=ans[i+1];
        }
    }
};
struct halfplane:public line
{
    double angle;
    halfplane(){}
    //表示向量 a->b逆时针(左侧)的半平面
    halfplane(point _a,point _b)
    {
        a=_a;
        b=_b;
    }
    halfplane(line v)
    {
        a=v.a;
        b=v.b;
    }
    void calcangle()
    {
        angle=atan2(b.y-a.y,b.x-a.x);
    }
    bool operator<(const halfplane &b)const
    {
        return angle<b.angle;
    }
};
```

```
struct halfplanes
{
    int n;
    halfplane hp[maxp];
    point p[maxp];
    int que[maxp];
    int st,ed;
    void push(halfplane tmp)
    {
        hp[n++]=tmp;
    }
    void unique()
    {
        int m=1,i;
        for (i=1;i<n;i++)
        {
            if (dblcmp(hp[i].angle-hp[i-1].angle))hp[m++]=hp[i];
            else if
(dblcmp(hp[m-1].b.sub(hp[m-1].a).det(hp[i].a.sub(hp[m-1].a))>0))hp[m-1]=hp[i];
        }
        n=m;
    }
    bool halfplaneinsert()
    {
        int i;
        for (i=0;i<n;i++)hp[i].calcangle();
        sort(hp,hp+n);
        unique();
        que[st=0]=0;
        que[ed=1]=1;
        p[1]=hp[0].crosspoint(hp[1]);
        for (i=2;i<n;i++)
        {
            while
(st<ed&&dblcmp((hp[i].b.sub(hp[i].a).det(p[ed].sub(hp[i].a))))<0)ed--;
            while (st<ed&&dblcmp((hp[i].b.sub(hp[i].a).det(p[st
+1].sub(hp[i].a))))<0)st++;
            que[++ed]=i;
            if (hp[i].parallel(hp[que[ed-1]]))return false;
            p[ed]=hp[i].crosspoint(hp[que[ed-1]]);
        }
        while
(st<ed&&dblcmp(hp[que[st]].b.sub(hp[que[st]].a).det(p[ed].sub(hp[que[st]].a)))<0
)ed--;
        while (st<ed&&dblcmp(hp[que[ed]].b.sub(hp[que[ed]].a).det(p[st
+1].sub(hp[que[ed]].a)))<0)st++;
        if (st+1>=ed)return false;
        return true;
    }
    void getconvex(polygon &con)
    {
        p[st]=hp[que[st]].crosspoint(hp[que[ed]]);
        con.n=ed-st+1;
        int j=st,i=0;
        for (;j<=ed;i++,j++)
        {
```

```
            con.p[i]=p[j];
        }
    }
};
struct point3
{
    double x,y,z;
    point3(){}
    point3(double _x,double _y,double _z):
    x(_x),y(_y),z(_z){};
    void input()
    {
        scanf("%lf%lf%lf",&x,&y,&z);
    }
    void output()
    {
        printf("%.2lf %.2lf %.2lf\n",x,y,z);
    }
    bool operator==(point3 a)
    {
        return dblcmp(a.x-x)==0&&dblcmp(a.y-y)==0&&dblcmp(a.z-z)==0;
    }
    bool operator<(point3 a)const
    {
        return dblcmp(a.x-x)==0?dblcmp(y-a.y)==0?dblcmp(z-a.z)<0:y<a.y:x<a.x;
    }
    double len()
    {
        return sqrt(len2());
    }
    double len2()
    {
        return x*x+y*y+z*z;
    }
    double distance(point3 p)
    {
        return sqrt((p.x-x)*(p.x-x)+(p.y-y)*(p.y-y)+(p.z-z)*(p.z-z));
    }
    point3 add(point3 p)
    {
        return point3(x+p.x,y+p.y,z+p.z);
    }
    point3 sub(point3 p)
    {
        return point3(x-p.x,y-p.y,z-p.z);
    }
    point3 mul(double d)
    {
        return point3(x*d,y*d,z*d);
    }
    point3 div(double d)
    {
        return point3(x/d,y/d,z/d);
    }
    double dot(point3 p)
    {
```

```
        return x*p.x+y*p.y+z*p.z;
    }
    point3 det(point3 p)
    {
        return point3(y*p.z-p.y*z,p.x*z-x*p.z,x*p.y-p.x*y);
    }
    double rad(point3 a,point3 b)
    {
        point3 p=(*this);
        return acos(a.sub(p).dot(b.sub(p))/(a.distance(p)*b.distance(p)));
    }
    point3 trunc(double r)
    {
        r/=len();
        return point3(x*r,y*r,z*r);
    }
    point3 rotate(point3 o,double r)
    {
    }
};
struct line3
{
    point3 a,b;
    line3(){}
    line3(point3 _a,point3 _b)
    {
        a=_a;
        b=_b;
    }
    bool operator==(line3 v)
    {
        return (a==v.a)&&(b==v.b);
    }
    void input()
    {
        a.input();
        b.input();
    }
    double length()
    {
        return a.distance(b);
    }
    bool pointonseg(point3 p)
    {
        return
dblcmp(p.sub(a).det(p.sub(b)).len())==0&&dblcmp(a.sub(p).dot(b.sub(p)))<=0;
    }
    double dispointtoline(point3 p)
    {
        return b.sub(a).det(p.sub(a)).len()/a.distance(b);
    }
    double dispointtoseg(point3 p)
    {
        if (dblcmp(p.sub(b).dot(a.sub(b)))<0||dblcmp(p.sub(a).dot(b.sub(a)))<0)
        {
            return min(p.distance(a),p.distance(b));
```

```
        }
        return dispointtoline(p);
    }
    point3 lineprog(point3 p)
    {
        return a.add(b.sub(a).trunc(b.sub(a).dot(p.sub(a))/b.distance(a)));
    }
    point3 rotate(point3 p,double ang)//p绕此向量逆时针arg角度
    {
        if (dblcmp((p.sub(a).det(p.sub(b)).len()))==0)return p;
        point3 f1=b.sub(a).det(p.sub(a));
        point3 f2=b.sub(a).det(f1);
        double len=fabs(a.sub(p).det(b.sub(p)).len()/a.distance(b));
        f1=f1.trunc(len);f2=f2.trunc(len);
        point3 h=p.add(f2);
        point3 pp=h.add(f1);
        return
h.add((p.sub(h)).mul(cos(ang*1.0))).add((pp.sub(h)).mul(sin(ang*1.0)));
    }
};
struct plane
{
    point3 a,b,c,o;
    plane(){}
    plane(point3 _a,point3 _b,point3 _c)
    {
        a=_a;
        b=_b;
        c=_c;
        o=pvec();
    }
    plane(double _a,double _b,double _c,double _d)
    {
        //ax+by+cz+d=0
        o=point3(_a,_b,_c);
        if (dblcmp(_a)!=0)
        {
            a=point3((-_d-_c-_b)/_a,1,1);
        }
        else if (dblcmp(_b)!=0)
        {
            a=point3(1,(-_d-_c-_a)/_b,1);
        }
        else if (dblcmp(_c)!=0)
        {
            a=point3(1,1,(-_d-_a-_b)/_c);
        }
    }
    void input()
    {
        a.input();
        b.input();
        c.input();
        o=pvec();
    }
    point3 pvec()
```

```
{
    return b.sub(a).det(c.sub(a));
}
bool pointonplane(point3 p)//点是否在平面上
{
    return dblcmp(p.sub(a).dot(o))==0;
}
//0 不在
//1 在边界上
//2 在内部
int pointontriangle(point3 p)//点是否在空间三角形abc上
{
    if (!pointonplane(p))return 0;
    double s=a.sub(b).det(c.sub(b)).len();
    double s1=p.sub(a).det(p.sub(b)).len();
    double s2=p.sub(a).det(p.sub(c)).len();
    double s3=p.sub(b).det(p.sub(c)).len();
    if (dblcmp(s-s1-s2-s3))return 0;
    if (dblcmp(s1)&&dblcmp(s2)&&dblcmp(s3))return 2;
    return 1;
}
//判断两平面关系
//0 相交
//1 平行但不重合
//2 重合
bool relationplane(plane f)
{
    if (dblcmp(o.det(f.o).len()))return 0;
    if (pointonplane(f.a))return 2;
    return 1;
}
double angleplane(plane f)//两平面夹角
{
    return acos(o.dot(f.o)/(o.len()*f.o.len()));
}
double dispoint(point3 p)//点到平面距离
{
    return fabs(p.sub(a).dot(o)/o.len());
}
point3 pttoplane(point3 p)//点到平面最近点
{
    line3 u=line3(p,p.add(o));
    crossline(u,p);
    return p;
}
int crossline(line3 u,point3 &p)//平面和直线的交点
{
    double x=o.dot(u.b.sub(a));
    double y=o.dot(u.a.sub(a));
    double d=x-y;
    if (dblcmp(fabs(d))==0)return 0;
    p=u.a.mul(x).sub(u.b.mul(y)).div(d);
    return 1;
}
int crossplane(plane f,line3 &u)//平面和平面的交线
```

```
    {
        point3 oo=o.det(f.o);
        point3 v=o.det(oo);
        double d=fabs(f.o.dot(v));
        if (dblcmp(d)==0)return 0;
        point3 q=a.add(v.mul(f.o.dot(f.a.sub(a))/d));
        u=line3(q,q.add(oo));
        return 1;
    }
};
polygon a,b;
int main()
{
    int i,j,k;
    a.n=b.n=4;
    double w,h,g;
    cin>>w>>h>>g;
    g/=180.0;
    g*=pi;
    a.p[0]=point(-w/2.0,-h/2.0);
    a.p[1]=point(w/2.0,-h/2.0);
    a.p[2]=point(w/2.0,h/2.0);
    a.p[3]=point(-w/2.0,h/2.0);
    for (i=0;i<4;i++)
    {
        b.p[i]=(a.p[i]).rotate(point(0,0),g);
    }
    polygons p;
    p.push(a);p.push(b);
    printf("%.12lf\n",p.polyareaunion());
    return 0;
}
```