#include<cstdio>

#include<cmath>

#include<cstring>

#include<algorithm>

#define sqr(x) ((x)\*(x))

using namespace std;

const double pi=acos(-1.0);

const double eps=1e-8;

int sgn(double x)

{

return x<-eps?-1:x>eps;

}

struct point

{

double x,y;

point() {}

point(double a,double b): x(a),y(b) {}

void read()

{

scanf("%lf%lf",&x,&y);

}

void show()

{

printf("(%lf,%lf)\n",x,y);

}

point operator + (const point &b) const

{

return point(x+b.x,y+b.y);

}

point operator - (const point &b) const

{

return point(x-b.x,y-b.y);

}

point operator \* (const double &b) const

{

return point(x\*b,y\*b);

}

point operator / (const double &b) const

{

return point(x/b,y/b);

}

bool operator == (const point &b) const

{

return (sgn(y-b.y)==0&&sgn(x-b.x)==0);

}

bool operator < (const point &b) const

{

if(sgn(y-b.y)==0)

return x<b.x;

return y<b.y;

}

double operator \* (const point &b) const

{

return x\*b.x+y\*b.y;

}

double X (const point &b) const

{

return x\*b.y-y\*b.x;

}

point rot (const double &ta) const

{

return point(cos(ta)\*x-sin(ta)\*y,sin(ta)\*x+cos(ta)\*y);

}

point rot90 (const double &d) const

{

return point(-y,x) / len()\*d;

}

point unit()

{

return point(x,y)/len();

}

double dis (const point &b) const

{

return sqrt(fabs((x-b.x)\*(x-b.x)+(y-b.y)\*(y-b.y)));

}

double dis2 (const point &b) const

{

return fabs(sqr(x-b.x)+sqr(y-b.y));

}

double len() const

{

return sqrt(x\*x+y\*y);

}

double len2() const

{

return x\*x+y\*y;

}

};

struct line

{

point s,e;

double ang;

line () {}

line(point a,point b): s(a),e(b),ang(atan2(e.y-s.y,e.x-s.x)) {}

bool operator / (const line &l) const

{

return sgn((l.e-l.s).X(e-s))==0;

}

bool operator == (const line &l) const

{

return sgn((l.s-s).X(l.e-s))==0&&sgn((l.s-e).X(l.e-e))==0;

}

bool operator < (const line &b) const

{

if(sgn(b.ang-ang)!=0)return ang<b.ang;

return sgn((b.e-b.s).X(e-b.s))>=0;

}

bool isright(const point &p) const

{

return sgn((p-s).X(e-s))>0;

}

bool ispointonline(const point &a) const

{

return sgn((a-s).X(s-e))==0;

}

bool ispointonseg(const point &a) const

{

return sgn((a-s).X(s-e))==0&&sgn((a-s)\*(a-e))<=0;

}

bool iscross(const line &l) const //严格相交

{

int t1=sgn((l.s-s).X(e-s));

int t2=sgn((l.e-s).X(e-s));

int t3=sgn((s-l.s).X(l.e-l.s));

int t4=sgn((e-l.s).X(l.e-l.s));

return t1\*t2<0&&t3\*t4<0;

}

point crosspoint(const line &l)

{

double a1 = (s-l.s).X(e-l.s), a2 = -(s-l.e).X(e-l.e);

return (l.s \* a2 + l.e \* a1) / (a1 + a2);

}

point ref(const point &p) const

{

point pj =s+(e - s)\*((e - s)\*(p - s)/(e - s).len2());

return pj \* 2 - p;

}

double dis(const point &p) const

{

return fabs((e-s).X(p-s)/(e-s).len());

}

};

struct polygon

{

point p[1010];

int n;

double area()

{

double ans=0;

for(int i=1;i<=n;i++)

ans+=p[i-1].X(p[i%n]);

return ans/2;

}

//任意多边形重心= =

point center()

{

double sx=0,sy=0,sa=0;

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

{

point p0=p[0];

point p1=p[i-1];

point p2=p[i];

double cx=p0.x+p1.x+p2.x;

double cy=p0.y+p1.y+p2.y;

double A=(p1-p0).X(p2-p0);

sa+=A;

sx+=cx\*A;

sy+=cy\*A;

}

return point(sx/sa/3,sy/sa/3);

}

void grahanm(polygon &ans)

{

sort(p,p+n);

ans.p[0]=p[0];ans.p[1]=p[1];

int now=2;

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

{

while(now>1&&sgn((ans.p[now-1]-p[i]).X(ans.p[now-2]-p[i]))>=0)now--;

ans.p[now++]=p[i];

}

int top=now;

for(int i=n-2;i>=0;i--)

{

while(now>top&&sgn((ans.p[now-1]-p[i]).X(ans.p[now-2]-p[i]))>=0)now--;

ans.p[now++]=p[i];

}

ans.n=--now;

}

bool isin(point a)

{

a=a-p[0];

if(sgn((p[1]-p[0]).X(a))<0)return 0;

int l=1,r=n-1;

while(l<r)

{

int mid=(l+r)/2+1;

double temp=(p[mid]-p[0]).X(a);

if(sgn(temp)==0)

{

line l1=line(p[0],p[mid]);

return l1.ispointonseg(a+p[0]);

}

if(temp>0)l=mid;

else r=mid-1;

}

if(l==n-1)return 0;

return sgn((p[l+1]-p[l]).X(a+p[0]-p[l]))>=0;

}

void cut(point p1,point p2)

{

p[n]=p[0];

polygon q;

line l1, l2;

int t1, t2;

q.n=0;

for(int i=0;i<n;i++)

{

t1 =sgn((p2-p1).X(p[i]-p1));

t2 =sgn((p2-p1).X(p[i+1]-p1));

if(t1>=0)q.p[q.n++]=p[i];

if(t1\*t2<0)

{

l1=line(p1, p2);

l2=line(p[i],p[i + 1]);

q.p[q.n++]=l1.crosspoint(l2);

}

}

n=q.n;

for(int i=0;i<q.n;i++)p[i]=q.p[i];

p[n]=p[0];

}

};

//最小包围矩形

void solve()

{

double minarea=1e20,minlen=1e20;

int l=1,r=1,u=1;

a.p[a.n]=a.p[0];

for(int i=0;i<a.n;i++)

{

point vec=(a.p[i+1]-a.p[i]).unit();

while((a.p[(r+1)%a.n]-a.p[i])\*vec>(a.p[r%a.n]-a.p[i])\*vec)r++;

while(u<r||vec.X(a.p[(u+1)%a.n]-a.p[i])>vec.X(a.p[u%a.n]-a.p[i]))u++;

while(l<u||(a.p[(l+1)%a.n]-a.p[i])\*vec<(a.p[l%a.n]-a.p[i])\*vec)l++;

double aa=(a.p[r%a.n]-a.p[i])\*vec-(a.p[l%a.n]-a.p[i])\*vec;

double bb=vec.X(a.p[u%a.n]-a.p[i]);

minarea=min(minarea,aa\*bb);

minlen=min(minlen,(aa+bb)\*2);

}

printf("%.2f %.2f\n",minarea,minlen);

}

//点是否在多边形内部

int isin(point p,point a[],int n)

{

int cnt=0;

for(int i=0;i<n;i++)

if(line(a[i],a[(i+1)%n]).ispointonseg(p))return 0;

for(int i=0;i<n;i++)

{

point s=a[i],e=a[(i+1)%n];

if(sgn(s.y-e.y)==0)continue;

if(s.y>e.y)swap(s,e);

if(sgn(s.y-p.y)<0&&sgn(e.y-p.y)>0&&(e-s).X(p-s)>0)cnt++;

if(sgn(s.y-p.y)==0&&sgn(s.x-p.x)>0)cnt++;

}

if(cnt&1)return 1;

return -1;

}

// 半径r的圆是否能放到简单多边形内 n^3!

bool check(point p)

{

if(isin(p)!=1)return 0;

for(int i=0;i<n;i++)

if(sgn(a[i].dis(p)-r)<0)return 0;

for(int i=0;i<n;i++)

{

line l=line(a[i],a[i+1]);

double d=l.dis(p);

if(sgn(d-r)>=0)continue;

point o=(l.ref(p)+p)/2;

if(sgn((o-a[i])\*(o-a[i+1]))<0)return 0;

}

return 1;

}

bool solve()

{

for(int i=0;i<n;i++)

for(int j=i+1;j<n;j++)

{

line l1,l2;

point vec1=(a[i+1]-a[i]).rot90().unit()\*r;

point vec2=(a[j+1]-a[j]).rot90().unit()\*r;

for(int d1=0;d1<2;d1++)

for(int d2=0;d2<2;d2++)

{

if(d1)l1=line(a[i]-vec1,a[i+1]-vec1);

else l1=line(a[i]+vec1,a[i+1]+vec1);

if(d2)l2=line(a[j]-vec2,a[j+1]-vec2);

else l2=line(a[j]+vec2,a[j+1]+vec2);

if(l1/l2)continue;

point o=l1.crosspoint(l2);

if(check(o))return 1;

}

}

for(int i=0;i<n;i++)

for(int j=i+1;j<n;j++)

{

double d=a[i].dis(a[j]);

if(sgn(d-2\*r)>0)continue;

d=sqrt(r\*r-d\*d/4);

point mid=(a[i]+a[j])/2;

point vec=(a[i]-a[j]).rot90().unit()\*d;

if(check(mid+vec))return 1;

if(check(mid-vec))return 1;

}

for(int i=0;i<n;i++)

for(int j=0;j<n;j++)

{

point vec=(a[i+1]-a[i]).rot90().unit()\*r;

line l;

for(int d=0;d<2;d++)

{

if(d)l=line(a[i]+vec,a[i+1]+vec);

else l=line(a[i]-vec,a[i+1]-vec);

double d=l.dis(a[j]);

if(sgn(d-r)>0)continue;

point o=(l.ref(a[j])+a[j])/2;

d=sqrt(r\*r-d\*d);

point dir=(a[i+1]-a[i]).unit()\*d;

if(check(o+dir))return 1;

if(check(o-dir))return 1;

}

}

return 0;

}

//多圆面积并

struct cir

{

point o;

double r;

};

struct nima

{

double ta;

int num;

bool operator < (const nima &b) const

{

if(sgn(ta-b.ta)==0)

return num>b.num;

return ta<b.ta;

}

}gao[10000];

void add(cir a,cir b,int &cnt)

{

double d=a.o.dis(b.o);

if(sgn(d-(b.r-a.r))<=0)

{

gao[cnt].ta=0;gao[cnt++].num=1;

gao[cnt].ta=2\*pi;gao[cnt++].num=-1;

return;

}

if(sgn(d-a.r-b.r)>=0||sgn(d-(a.r-b.r))<=0)return;

d=acos((sqr(a.r)+sqr(d)-sqr(b.r))/2/a.r/d);

double t1=atan2(b.o.y-a.o.y,b.o.x-a.o.x)-d;if(sgn(t1)<0)t1+=2\*pi;

double t2=atan2(b.o.y-a.o.y,b.o.x-a.o.x)+d;if(sgn(t2-2\*pi)>=0)t2-=2\*pi;if(sgn(t2)<0)t2+=2\*pi;

if(sgn(t1-t2)<=0)

{

gao[cnt].ta=t1;

gao[cnt++].num=1;

gao[cnt].ta=t2;

gao[cnt++].num=-1;

}

else

{

gao[cnt].ta=t1;

gao[cnt++].num=1;

gao[cnt].ta=2\*pi;

gao[cnt++].num=-1;

gao[cnt].ta=0;

gao[cnt++].num=1;

gao[cnt].ta=t2;

gao[cnt++].num=-1;

}

}

int n;

cir a[1010];

double ans[1010];

void solve()

{

for(int i=0;i<1010;i++)ans[i]=0;

for(int i=1;i<=n;i++)

{

int gn=0;

for(int j=1;j<=n;j++)

add(a[i],a[j],gn);

sort(gao,gao+gn);

int now=0;

point cao1=point(cos(gao[0].ta),sin(gao[0].ta))\*a[i].r+a[i].o;

point cao2=point(cos(gao[1].ta),sin(gao[1].ta))\*a[i].r+a[i].o;

for(int j=0;j<gn-1;j++)

{

now+=gao[j].num;

double cao=gao[j+1].ta-gao[j].ta;

ans[now]+=sqr(a[i].r)\*(cao-sin(cao));

ans[now]+=cao1.X(cao2);

cao1=cao2;

cao2=point(cos(gao[j+2].ta),sin(gao[j+2].ta))\*a[i].r+a[i].o;

}

}

for(int i=1;i<=n;i++)ans[i]=(ans[i]-ans[i+1])/2;

}

//最小圆覆盖

point a[510];

int main()

{

int n;

while(scanf("%d",&n),n)

{

for(int i=1;i<=n;i++)

scanf("%lf%lf",&a[i].x,&a[i].y);

random\_shuffle(a+1,a+1+n);

point o=a[1];

double r=0;

for(int i=1;i<=n;i++)

if(sgn(a[i].dis(o)-r)>0)

{

o=a[i];r=0;

for(int j=1;j<i;j++)

if(sgn(a[j].dis(o)-r)>0)

{

o=(a[i]+a[j])/2;

r=a[j].dis(o);

for(int k=1;k<j;k++)

if(sgn(a[k].dis(o)-r)>0)

{

o=cal(a[i],a[j],a[k]);//外接圆圆心

r=a[k].dis(o);

}

}

}

printf("%.2f %.2f %.2f\n",o.x,o.y,r);

}

}

//(x,y,z)向量射进眼睛 逆时针转

ma ROTATE()

{

double x,y,z,ta;

scanf("%lf%lf%lf%lf",&x,&y,&z,&ta);

ta=ta/180\*pi;

ma ans;

ans.init();

double len = sqrt(x \* x + y \* y + z \* z);

x /= len;

y /= len;

z /= len;

ans.a[0][0] = x \* x + (1 - x \* x) \* cos(ta);

ans.a[0][1] = x \* y \* (1 - cos(ta)) - z \* sin(ta);

ans.a[0][2] = x \* z \* (1 - cos(ta)) + y \* sin(ta);

ans.a[1][0] = y \* x \* (1 - cos(ta)) + z \* sin(ta);

ans.a[1][1] = y \* y + (1 - y \* y) \* cos(ta);

ans.a[1][2] = y \* z \* (1 - cos(ta)) - x \* sin(ta);

ans.a[2][0] = z \* x \* (1 - cos(ta)) - y \* sin(ta);

ans.a[2][1] = z \* y \* (1 - cos(ta)) + x \* sin(ta);

ans.a[2][2] = z \* z + (1 - z \* z) \* cos(ta);

return ans;

}

//三维凸包

#include <cstdio>

#include <cstring>

#include <algorithm>

#include <vector>

#include <string>

#include <map>

#include <set>

#include <queue>

#include <cmath>

#include <iostream>

#include <cstdlib>

#include <sstream>

using namespace std;

const int MAXN=10001;

const double eps=1e-8;

struct \_3DCH

{

struct point

{

double x, y, z;

point() {}

point(double \_x, double \_y, double \_z): x(\_x), y(\_y), z(\_z) {}

void read()

{

scanf("%lf %lf %lf",&x,&y,&z);

}

point operator - (const point p1)

{

return point(x - p1.x, y - p1.y, z - p1.z);

}

point operator \* (point p)

{

return point(y\*p.z-z\*p.y, z\*p.x-x\*p.z, x\*p.y-y\*p.x);

}

double operator ^ (point p)

{

return x\*p.x+y\*p.y+z\*p.z;

}

};

struct fac

{

int a, b, c;

bool ok;

};

int n;

point P[MAXN];

int cnt;

fac F[MAXN\*8];

map < pair<int,int> ,int > to;

double totalvolume;

double vlen(point a)

{

return sqrt(a.x\*a.x+a.y\*a.y+a.z\*a.z);

}

double area(point a, point b, point c)

{

return vlen((b-a)\*(c-a));

}

double volume(point a, point b, point c)

{

return (b-a)\*(c-a)^a;

}

double pointof(point &p, fac &f)

{

point m = P[f.b]-P[f.a], n = P[f.c]-P[f.a], t = p-P[f.a];

return (m \* n) ^ t;

}

void deal(int p, int a, int b)

{

int f = to[make\_pair(a,b)];

fac add;

if (F[f].ok)

{

if (pointof(P[p], F[f]) > eps)

dfs(p, f);

else

{

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

to[make\_pair(p,b)] = to[make\_pair(a,p)] = to[make\_pair(b,a)] = cnt;

F[cnt++] = add;

totalvolume+=volume(P[F[cnt-1].a],P[F[cnt-1].b],P[F[cnt-1].c]);

}

}

}

void dfs(int p, int cur)

{

F[cur].ok = 0;

totalvolume-=volume(P[F[cur].a],P[F[cur].b],P[F[cur].c]);

deal(p, F[cur].b, F[cur].a);

deal(p, F[cur].c, F[cur].b);

deal(p, F[cur].a, F[cur].c);

}

void init()

{

n=cnt=0;

totalvolume=0;

}

bool add(point p)

{

P[n++]=p;

if(n<4)return 0;

if(!cnt)

{

bool sb = 1;

for (int i = 1; i < n; i++)

{

if (vlen(P[0] - P[i]) > eps)

{

swap(P[1], P[i]);

sb = 0;

break;

}

}

if (sb)return 0;

sb = 1;

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

{

if (vlen((P[0] - P[1]) \* (P[1] - P[i])) > eps)

{

swap(P[2], P[i]);

sb = 0;

break;

}

}

if (sb)return 0;

sb = 1;

for (int i = 3; i < n; i++)

{

if (fabs((P[0] - P[1]) \* (P[1] - P[2]) ^ (P[0] - P[i])) > eps)

{

swap(P[3], P[i]);

sb = 0;

break;

}

}

if (sb)return 0;

fac add;

for (int i = 0; i < 4; i++)

{

add.a = (i+1)%4, add.b = (i+2)%4, add.c = (i+3)%4, add.ok = 1;

if (pointof(P[i], add) > 0)

swap(add.b, add.c);

to[make\_pair(add.a,add.b)] = to[make\_pair(add.b,add.c)] = to[make\_pair(add.c,add.a)] = cnt;

F[cnt++] = add;

totalvolume+=volume(P[F[cnt-1].a],P[F[cnt-1].b],P[F[cnt-1].c]);

}

for(int i=4;i<n;i++)

{

for (int j = 0; j < cnt; j++)

if (F[j].ok && pointof(P[i], F[j]) > eps)

{

dfs(i, j);

break;

}

}

return 1;

}

else

{

for (int j = 0; j < cnt; j++)

if (F[j].ok && pointof(P[n-1], F[j]) > eps)

{

dfs(n-1, j);

return 1;

}

return 0;

}

}

void solve(int \_n)

{

init();

double ans=0;

int id=1;

for(int i=1;i<=\_n;i++)

{

point now;now.read();

totalvolume=0;

if(add(now)&&(totalvolume-ans)>eps)

{

id=i;

ans=totalvolume;

}

}

if(ans<eps)ans=0;

printf("%d %.2f\n",id,ans/6);

}

}gao;

int main()

{

int n,cas=0;

while(~scanf("%d",&n))

{

printf("Case #%d:\n",++cas);

gao.solve(n);

}

}

#include <iostream>

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <cmath>

#include <set>

using namespace std;

bool wintess(long long a,int q,long long m,long long n)

{

long long x=exp\_mod(a,m,n),y;

for(int j=1;j<=q;j++)

{

y=mult\_mod(x,x,n);

if(y==1&&x!=1&&x!=(n-1))return 1;

x=y;

}

if(y!=1)return 1;

return 0;

}

bool miler(long long n,int T)

{

if(n==1)return 0;

if(n==2)return 1;

if(n%2==0)return 0;

long long m=n-1;

int q=0;

while((m&1)==0)m>>=1,q++;

for(int i=1;i<=T;i++)

{

long long a=rand()%(n-2)+2;

if(wintess(a,q,m,n))return 0;

}

return 1;

}

long long rho(long long n)

{

long long x=rand()%n,y=x;

int i=1,k=2;

long long t;

while(1)

{

i++;

x=mult\_mod(x,x,n)+1;

if(x>=n)x-=n;

t=gcd(y-x,n);

if(t>1&&t<n)

return t;

if(x==y)

return n;

if(i==k)

{

y=x;

k<<=1;

}

}

}

long long SQRT(long long n)

{

long long ans=max(1LL,(long long)sqrt((double)n)-3);

while((ans+1)\*(ans+1)<=n)ans++;

return ans;

}

long long SQUFOF(int k,long long n)

{

int i;

n = n \* k;

static long long temp,sqn,p[10001],q[10001],b[10001];

sqn = SQRT(n);

p[0] = sqn;

q[0] = 1;

q[1] = n - p[0]\*p[0];

if(q[1]==0)return 1;

for(i = 1; i < 10000; ++i)

{

b[i] = (sqn + p[i - 1])/q[i];

p[i] = b[i] \* q[i] - p[i - 1];

q[i + 1] = q[i - 1] + b[i]\*(p[i - 1] - p[i]);

temp = SQRT(q[i]);

if(temp \* temp == q[i])

break;

}

if(i==10000)return 1;

b[0] = ((sqn - p[i - 1])/temp);

p[0] = b[0]\*temp + p[i - 1];

q[0] = temp;

q[1] = (n - (p[0]\*p[0]))/q[0];

for(i = 1; i < 10000; ++i)

{

b[i] = (sqn + p[i - 1])/q[i];

p[i] = b[i] \* q[i] - p[i - 1];

q[i + 1] = q[i - 1] + b[i]\*(p[i - 1] - p[i]);

if(p[i - 1] == p[i])

break;

}

if(i==10000)return 1;

return gcd(n/k,p[i]);

}

set <long long> p;

void fj(long long n)

{

long long te,res = SQRT(n);

while(res \* res == n && n != 1)

{

n = res;

res = SQRT(n);

}

if(n==1)return;

if(miler(n,10))

{

p.insert(n);

return;

}

int k=1;

while(k<=10&&( (res = SQUFOF(k,n)) == 1 || res == n) ) ++k;

while(res==n||res==1||n%res!=0)res=rho(n);

fj(res);

fj(n/res);

}

#include<cstdio>

#include<cmath>

#include<cstring>

#include<algorithm>

using namespace std;

const int w=16;

const int M=(1<<w)-1;

int gcd(int a,int b){return b==0?a:gcd(b,a%b);}

void ext\_gcd(int a,int b,int &x,int &y)

{

if (!b){x=1,y=0;return;}

ext\_gcd(b,a%b,x,y);

int t=x;x=y,y=t-a/b\*y;

}

int inv(int a,int b,int n)

{

int x,y,e;

ext\_gcd(a,n,x,y);

e=(long long)x\*b%n;

return e<0?e+n:e;

}

int exp\_mod(int a,int b,int m)

{

int ans=1;

while(b)

{

if(b&1)

ans=(long long)ans\*a%m;

a=(long long)a\*a%m;

b>>=1;

}

return ans;

}

struct node

{

int k,num,nx;

}hash[100000];

int head[M+10],cnt;

int find(int x)

{

int k=x>>w;

for(int i=head[x&M];i!=-1;i=hash[i].nx)

if(hash[i].k==k)return hash[i].num;

return -1;

}

void insert(int x,int num)

{

if(find(x)!=-1)return;

hash[cnt].k=x>>w;

hash[cnt].num=num;

hash[cnt].nx=head[x&M];

head[x&M]=cnt++;

}

int babystep(int a,int y,int p)

{

for(int i=0;i<50;i++)if(exp\_mod(a,i,p)==y)return i;

int g=0,D=1,d;

while((d=gcd(a,p))!=1)

{

if(y%d)return -1;

y/=d;p/=d;g++;

D=(long long)a/d\*D%p;

}

int m=ceil(sqrt((double)p));

cnt=0;memset(head,-1,sizeof(head));

for(int i=0,cao=1;i<m;i++,cao=(long long)cao\*a%p)insert(cao,i);

for(int i=0,cao=exp\_mod(a,m,p);i<m;i++,D=(long long)D\*cao%p)

{

int t=inv(D,y,p);

int j=find(t);

if(j!=-1)return i\*m+j+g;

}

return -1;

}

int main()

{

int a,p,y;

while(scanf("%d%d%d",&a,&p,&y))

{

if(!a&&!p&&!y)break;

a%=p;y%=p;

int ans=babystep(a,y,p);

if(ans!=-1)

printf("%d\n",ans);

else

puts("No Solution");

}

}

long long FLOOR(long long A,long long B)

{

if(B<0){A=-A;B=-B;}

if(A%B==0||A>0)return A/B;

return A/B-1;

}

//sigma(A\*i+B)/n 0<=i<=a;

long long dfs(long long a,long long A,long long B,long long n)

{

if(a<0)return 0;

long long D=gcd(gcd(abs(A),abs(B)),n);A/=D;B/=D;n/=D;

if(A%n==0)

{

A/=n;

return a\*(a+1)/2\*A+(a+1)\*FLOOR(B,n);

}

if(FLOOR(A,n)!=0) return a\*(a+1)/2\*FLOOR(A,n)+dfs(a,A-FLOOR(A,n)\*n,B,n);

if(FLOOR(B,n)!=0) return (a+1)\*FLOOR(B,n)+dfs(a,A,B-FLOOR(B,n)\*n,n);

long long M=(A\*a+B)/n;

long long R=A\*a+B-M\*n;

return dfs(M-1,n,R,A)+M;

}

//1-N有几个素数

const int maxn=100000010;

const int maxN=30000000;

int prime[maxn/10],flag[maxn],num;

int a;

bool vis[maxN];

long long P(long long n)

{

long long ans=0;

for(int i=a+1;1LL\*prime[i]\*prime[i]<=n;i++)ans+=flag[n/prime[i]]-i+1;

return ans;

}

long long N;

vector <int> gao[1300];

long long dfs(long long s,int now)

{

long long S=s;if(S<0)S=-S;

if(now==0)return N/S;

if(N/S<prime[now+1])return 1;

if(N/S<maxN)

{

gao[now].push\_back(N/s);

return 0;

}

return dfs(s,now-1)-dfs(-s\*prime[now],now-1);

}

int tree[maxN];

int read(int pos)

{

int ans=0;

while(pos)ans+=tree[pos],pos-=pos&-pos;

return ans;

}

void insert(int pos)

{

while(pos<maxN)tree[pos]++,pos+=pos&-pos;

}

long long cal(long long n)

{

if(n<=1)return 0;

if(n<maxn)return flag[n];

N=n;

a=1;

while(1LL\*prime[a]\*prime[a]\*prime[a]<=n)a++;

for(int i=1;i<=a;i++)gao[i].clear();

long long ans=dfs(1,a)-P(n)+a-1;

memset(tree,0,sizeof(tree));

memset(vis,0,sizeof(vis));

for(int i=1;i<=a;i++)

{

for(int j=prime[i];j<maxN;j+=prime[i])

if(!vis[j])

insert(j),vis[j]=1;

for(int j=0;j<gao[i].size();j++)

{

if(gao[i][j]>0)

ans+=gao[i][j]-read(gao[i][j]);

else ans-=-gao[i][j]-read(-gao[i][j]);

}

}

return ans;

}

int main()

{

for(int i=2;i<maxn;i++)

{

if(!flag[i])prime[++num]=i;

for(int j=1;j<=num&&i\*prime[j]<maxn;j++)

{

flag[i\*prime[j]]=1;

if(i%prime[j]==0)break;

}

}

for(int i=2;i<maxn;i++)flag[i]=flag[i-1]+(!flag[i]);

long long n;

while(cin>>n)

{

double sta=clock();

cout<<cal(n)<<endl;

printf("%.10f\n",clock()-sta);

}

}

//FFT

struct Z

{

double re,im;

Z(): re(0),im(0) {}

Z(double a,double b): re(a),im(b) {}

Z operator + (const Z &b) const

{

return Z(re+b.re,im+b.im);

}

Z operator - (const Z &b) const

{

return Z(re-b.re,im-b.im);

}

Z operator \* (const Z &b) const

{

return Z(re\*b.re-im\*b.im,re\*b.im+im\*b.re);

}

Z inv()

{

return Z(re,-im);

}

};

int gao(int x,int n)

{

int ans=0;

for(int i=0;i<n;i++)

{

ans<<=1;

if(x&1)ans++;

x>>=1;

}

return ans;

}

void FFT(Z \*ff,int n,int m,Z \*ans)

{

for(int i=0;i<n;i++)ans[i]=ff[gao(i,m)];

for(int len=1;len<n;len<<=1)

{

Z wo=Z(cos(-pi/len),sin(-pi/len)),w,t1,t2;

for(int k=0;k\*len<n;k+=2)

{

w=Z(1,0);

for(int i=0;i<len;i++)

{

t1=ans[i+k\*len],t2=w\*ans[i+(k+1)\*len],w=w\*wo;

ans[i+k\*len]=t1+t2;

ans[i+(k+1)\*len]=t1-t2;

}

}

}

}

Z ff[4\*maxn],A[4\*maxn];

void FFT(int a[],int n,int b[],int m,double ans[])

{

int N=1,M=0;

while(N<n+m)N\*=2,M++;

for(int i=0;i<n;i++)ff[i].re=a[i];

for(int i=n;i<N;i++)ff[i].re=0;

for(int i=0;i<m;i++)ff[i].im=b[i];

for(int i=m;i<N;i++)ff[i].im=0;

FFT(ff,N,M,A);

A[N]=A[0];

Z w1,w2,w3;

for(int i=0;i<N;i++)

{

w3=A[N-i].inv();

w1=A[i]+w3;

w2=A[i]-w3;

w1=w1\*w2;

w1=w1.inv();

ff[i]=Z(-w1.im/4,w1.re/4);

}

FFT(ff,N,M,A);

for(int i=0;i<N;i++)ans[i]=A[i].re/N;

}

//sbt

const int maxn=10000;

struct Node

{

int size, key, value, sum;

Node \*c[2];

};

int N;

Node memo[maxn\*17], \*cur, \*tr[maxn], \*nil;

inline Node\* newNode(int key, int v)

{

cur->key = key;

cur->value = cur->sum = v;

cur->size = 1;

cur->c[0] = cur->c[1] = nil;

return cur++;

}

inline void init()

{

nil = cur = memo;

nil = newNode(0, 0);

nil->size = 0;

for (int i = 1; i <= N; ++i)

tr[i] = nil;

}

inline void update(Node\* t)

{

t->sum = t->c[0]->sum + t->c[1]->sum + t->value;

t->size = t->c[0]->size + t->c[1]->size + 1;

}

inline void rotate(Node\*& t, int f)

{

Node\* k = t->c[f ^ 1];

t->c[f ^ 1] = k->c[f];

k->c[f] = t;

update(t);

update(k);

t = k;

}

inline void keep(Node\*& t, int f)

{

if (t == nil) return;

else if (t->c[f]->c[f]->size > t->c[f ^ 1]->size)

rotate(t, f ^ 1);

else if (t->c[f]->c[f ^ 1]->size > t->c[f ^ 1]->size)

rotate(t->c[f], f), rotate(t, f ^ 1);

else

return;

for (int i = 0; i < 2; ++i) keep(t->c[i], i);

for (int i = 0; i < 2; ++i) keep(t, i);

}

inline void insert(Node\*& t, int key, int v)

{

if (t == nil)

t = newNode(key, v);

else if (t->key == key)

{

t->value += v;

update(t);

}

else

{

insert(t->c[key > t->key], key, v);

update(t);

keep(t, key > t->key);

}

}

inline int read(Node\* t, int key)

{

int ret=0;

while (t != nil)

{

if (t->key <= key)

{

ret += t->value + t->c[0]->sum;

t = t->c[1];

}

else

t = t->c[0];

}

}

// Simplex Method for Linear Programming

//

// m - number of (less than) inequalities

// n - number of variables

//

// C - (m+1) by (n+1) array of coefficients:

//

// row 0 - objective function coefficients

// row 1:m - less-than inequalities

//

// column 0:n-1 - inequality coefficients

// column n - inequality constants (0 for objective function)

//

// X[n] - result variables

//

// return value - maximum value of objective function

// (-inf for infeasible, inf for unbounded)

//

#define MAXM 400 // leave one extra

#define MAXN 400 // leave one extra

#define EPS 1e-9

#define INF 1.0/0.0

double A[MAXM][MAXN];

int basis[MAXM], out[MAXN];

void pivot(int m, int n, int a, int b) {

int i,j;

for (i=0;i<=m;i++) if (i!=a) for (j=0;j<=n;j++) if (j!=b) {

A[i][j] -= A[a][j] \* A[i][b] / A[a][b];

}

for (j=0;j<=n;j++) if (j!=b) A[a][j] /= A[a][b];

for (i=0;i<=m;i++) if (i!=a) A[i][b] = -A[i][b]/A[a][b];

A[a][b] = 1/A[a][b];

i = basis[a];

basis[a] = out[b];

out[b] = i;

}

double simplex(int m, int n, double C[][MAXN], double X[]) {

int i,j,ii,jj; // i,ii are row indexes; j,jj are column indexes

for (i=1;i<=m;i++) for (j=0;j<=n;j++) A[i][j] = C[i][j];

for (j=0;j<=n;j++) A[0][j] = -C[0][j];

for (i=0;i<=m;i++) basis[i] = -i;

for (j=0;j<=n;j++) out[j] = j;

for(;;) {

for (i=ii=1;i<=m;i++) {

if (A[i][n]<A[ii][n]

|| (A[i][n]==A[ii][n] && basis[i]<basis[ii]))

ii=i;

}

if (A[ii][n] >= -EPS) break;

for (j=jj=0;j<n;j++)

if (A[ii][j]<A[ii][jj]-EPS

|| (A[ii][j]<A[ii][jj]-EPS && out[i]<out[j]))

jj=j;

if (A[ii][jj] >= -EPS) return -INF;

pivot(m,n,ii,jj);

}

for(;;) {

for (j=jj=0;j<n;j++)

if (A[0][j]<A[0][jj]

|| (A[0][j]==A[0][jj] && out[j]<out[jj]))

jj=j;

if (A[0][jj] > -EPS) break;

for (i=1,ii=0;i<=m;i++)

if (A[i][jj]>EPS &&

(!ii || A[i][n]/A[i][jj]<A[ii][n]/A[ii][jj]-EPS ||

(A[i][n]/A[i][jj]<A[ii][n]/A[ii][jj]+EPS

&& basis[i]<basis[ii])))

ii=i;

if (A[ii][jj] <= EPS) return INF;

pivot(m,n,ii,jj);

}

for (j=0;j<n;j++) X[j] = 0;

for (i=1;i<=m;i++) if (basis[i] >= 0) X[basis[i]] = A[i][n];

return A[0][n];

}

void print(int m, int n, char \*msg) { // not used -- debug only

int i,j;

printf("%s\n",msg);

for(i=0;i<=m;i++) {

for (j=0;j<=m;j++) printf(" %10d",i==j);

for (j=0;j<=n;j++) printf(" %10g",A[i][j]);

printf("\n");

}

for (i=0;i<=m;i++) printf(" %10d",basis[i]);

for (j=0;j<n;j++) printf(" %10d",out[j]);

printf("\n");

}