\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*plantilla\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include<bits/stdc++.h>

using namespace std;

#define all(v) (v).begin(),(v).end()

#define pb(x) push\_back(x)

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

#define mp(x,y) make\_pair((x),(y))

#define fast\_io() ios\_base::sync\_with\_stdio(0);cin.tie(0);

#define fi first

#define se second

#define sz(v) ((int)v.size())

typedef pair<int,int> pii;

typedef vector<int> vi;

typedef long long ll;

typedef unsigned long long ull;

typedef long double ld;

int main(){

//fast\_io();

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BIG-INTEGER\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import java.math.BigInteger;

import java.util.Scanner;

/\*\*

\* Created by ADVANCE on 24/10/2015.

\*/

public class dasd {

public static void main(String[] args) {

Scanner s=new Scanner (System.in);

BigInteger num=s.nextBigInteger();

BigInteger[] factoriales= new BigInteger[1040];

BigInteger aux1=new BigInteger("1");

factoriales[0]=aux1;

for (int i=1;i<=520;i++){

BigInteger \_i= new BigInteger(Integer.toString(i));

factoriales[i]=factoriales[i-1].multiply(\_i);

}

for(int i=1;i<520;i++){

int res=num.compareTo((factoriales[2\*i].divide((factoriales[i].multiply(factoriales[i+1])))));

if(res==0){

System.out.println(i+2);

}

}

System.out.println(factoriales[5]);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import java.math.BigInteger;

import java.util.Scanner;

import java.io.\*;

import java.math.\*;

/\*\*

\* Created by ADVANCE on 24/10/2015.

\*/

public class dasd {

public static void main(String[] args) {

Scanner s=new Scanner (System.in);

BigInteger num1=s.nextBigInteger();

BigInteger num2=s.nextBigInteger();

BigInteger n=s.nextBigInteger();

BigInteger[] fib= new BigInteger[21];

fib[1]=num1;

fib[2]=num2;

for (int i=3;i<=20;i++){

BigInteger \_i= new BigInteger(Integer.toString(i));

fib[i]=((fib[i-1].multiply(fib[i-1]))).add(fib[i-2]);

}

System.out.println(fib[n.intValue()]);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BIT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*+++

const int INF=(1<<29);

struct FT{

int BIT[100002];

FT(){

for(int i=0;i<100002;i++) BIT[i]=INF;

}

void update(int n,int val){

while(n<=100002){

BIT[n]=min(val,BIT[n]);

n+=(n & -n);

}

}

int query(int n){

int mini=INF;

while(n){

mini=min(mini,BIT[n]);

n-=(n & - n);

}

return mini;

}

};

int main(){

fast\_io();

int t;cin>>t;

while(t>0){

t--;

FT BIT;

int n;cin>>n;

vector< pair<int,pii> > v;

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

int a,b,c;cin>>a>>b>>c;

v.pb( mp( a, mp( b,c ) ) );

}

sort(all(v));

int cont=0;

for(int i=0;i<sz(v);i++){

int mini=BIT.query(v[i].se.fi);

if(mini < v[i].se.se) cont++;

else BIT.update(v[i].se.fi,v[i].se.se);

}

cout<<(n-cont)<<endl;

}

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GEOMETRY\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#define EPS 1e-8

#define PI acos(-1)

#define Vector Point

struct Point

{

double x, y;

Point(){}

Point(double a, double b) { x = a; y = b; }

double mod2() { return x\*x + y\*y; }

double mod() { return sqrt(x\*x + y\*y); }

double arg() { return atan2(y, x); }

Point ort() { return Point(-y, x); }

Point unit() { double k = mod(); return Point(x/k, y/k); }

};

Point operator +(const Point &a, const Point &b) { return Point(a.x + b.x, a.y + b.y); }

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

Point operator /(const Point &a, double k) { return Point(a.x/k, a.y/k); }

Point operator \*(const Point &a, double k) { return Point(a.x\*k, a.y\*k); }

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

bool operator !=(const Point &a, const Point &b){ return !(a==b);}

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

double dist(const Point &A, const Point &B) { return hypot(A.x - B.x, A.y - B.y); }

double cross(const Vector &A, const Vector &B) { return A.x \* B.y - A.y \* B.x; }

double dot(const Vector &A, const Vector &B) { return A.x \* B.x + A.y \* B.y; }

double area(const Point &A, const Point &B, const Point &C) { return cross(B - A, C - A); }

double get\_angle( Point A , Point P , Point B )

{

double ang = (A-P).arg() - (B-P).arg();

while(ang < 0) ang += 2\*PI;

while(ang > 2\*PI) ang -= 2\*PI;

return min(ang, 2\*PI-ang);

}

bool isInt(double k){ return abs(k - int(k + 0.5)) < 1e-5;}

/////

//responde si un punto esta en un poligono convexo incluyendo la frontera

bool isIn( Point O , Point A , Point B , Point X )

{

return abs( abs( area(O,A,B)) -( abs(area(O,A,X)) + abs(area(A,B,X)) + abs(area(O,B,X)) ) ) < EPS ;

}

// transforma el angulo de atan2 al rango 0 - 2PI

double f( double a )

{

if( a < 0 )a += 2\*PI;

return a;

}

////////////////////////////////////////////////////////////////////////////////

// Heron triangulo y cuadrilatero ciclico

//http://en.wikipedia.org/wiki/Brahmagupta's\_formula ----- sqrt((p-a) \* (p-b) \* (p-c)\*(p-d))

// http://mathworld.wolfram.com/CyclicQuadrilateral.html

// http://www.spoj.pl/problems/QUADAREA/

//adicional existencia de trapezoide y altura

//http://en.wikipedia.org/wiki/Trapezoid

double areaHeron(double a, double b, double c)

{

double s = (a + b + c) / 2;

return sqrt(s \* (s-a) \* (s-b) \* (s-c));

}

double circumradius(double a, double b, double c) { return a \* b \* c / (4 \* areaHeron(a, b, c)); }

double areaHeron(double a, double b, double c, double d)

{

double s = (a + b + c + d) / 2;

return sqrt((s-a) \* (s-b) \* (s-c) \* (s-d));

}

double circumradius(double a, double b, double c, double d) { return sqrt((a\*b + c\*d) \* (a\*c + b\*d) \* (a\*d + b\*c)) / (4 \* areaHeron(a, b, c, d)); }

//### DETERMINA SI P PERTENECE AL SEGMENTO AB ###########################################

bool between(const Point &A, const Point &B, const Point &P)

{

return P.x + EPS >= min(A.x, B.x) && P.x <= max(A.x, B.x) + EPS &&

P.y + EPS >= min(A.y, B.y) && P.y <= max(A.y, B.y) + EPS;

}

bool onSegment(const Point &A, const Point &B, const Point &P)

{

return abs(area(A, B, P)) < EPS && between(A, B, P);

}

//### DETERMINA SI EL SEGMENTO P1Q1 SE INTERSECTA CON EL SEGMENTO P2Q2 #####################

//11343\_UVA

bool intersects(const Point &P1, const Point &P2, const Point &P3, const Point &P4)

{

double A1 = area(P3, P4, P1);

double A2 = area(P3, P4, P2);

double A3 = area(P1, P2, P3);

double A4 = area(P1, P2, P4);

if( ((A1 > 0 && A2 < 0) || (A1 < 0 && A2 > 0)) &&

((A3 > 0 && A4 < 0) || (A3 < 0 && A4 > 0)))

return true;

else if(A1 == 0 && onSegment(P3, P4, P1)) return true;

else if(A2 == 0 && onSegment(P3, P4, P2)) return true;

else if(A3 == 0 && onSegment(P1, P2, P3)) return true;

else if(A4 == 0 && onSegment(P1, P2, P4)) return true;

else return false;

}

//### DETERMINA SI A, B, M, N PERTENECEN A LA MISMA RECTA ##############################

bool sameLine(Point P1, Point P2, Point P3, Point P4)

{

return area(P1, P2, P3) == 0 && area(P1, P2, P4) == 0;

}

//### SI DOS SEGMENTOS O RECTAS SON PARALELOS ###################################################

bool isParallel(const Point &P1, const Point &P2, const Point &P3, const Point &P4)

{

return abs( cross(P2 - P1, P4 - P3) ) <= EPS;

}

//### PUNTO DE INTERSECCION DE DOS RECTAS NO PARALELAS #################################

Point lineIntersection(const Point &A, const Point &B, const Point &C, const Point &D)

{

return A + (B - A) \* (cross(C - A, D - C) / cross(B - A, D - C));

}

Point circumcenter(const Point &A, const Point &B, const Point &C)

{

return (A + B + (A - B).ort() \* dot(C - B, A - C) / cross(A - B, A - C)) / 2;

}

//### FUNCIONES BASICAS DE POLIGONOS ################################################

bool isConvex(const vector <Point> &P)

{

int nP = P.size(), pos = 0, neg = 0;

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

{

double A = area(P[i], P[(i+1)%nP], P[(i+2)%nP]);

if(A < 0) neg++;

else if(A > 0) pos++;

}

return neg == 0 || pos == 0;

}

double area(const vector <Point> &P)

{

int nP = P.size();

double A = 0;

for(int i=1; i<=nP-2; i++)

A += area(P[0], P[i], P[i+1]);

return abs(A/2);

}

//### DETERMINA SI A ESTA EN EL INTERIOR DEL POLIGONO( sin boundary ) ########################

// works in simple poly

bool pointInPoly(const vector <Point> &P, const Point &A)

{

int nP = P.size(), cnt = 0;

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

{

int inf = i , sup = ( i + 1 )%nP;

if( P[ inf ].y > P[ sup ].y ) swap( inf , sup );

if( P[ inf ].y <= A.y && A.y < P[ sup ].y )

if( area( A , P[ inf ] , P[ sup ] ) > 0 )

cnt++;

}

return (cnt % 2) == 1;

}

/\* TEOREMA DE PICK

A = I + B/2 - 1, donde:

A = Area de un poligono de coordenadas enteras

I = Numero de puntos enteros en su interior

B = Numero de puntos enteros sobre sus bordes

Haciendo un cambio en la formula : I=(2A-B+2)/2, tenemos una forma de calcular

el numero de puntos enteros en el interior del poligono

int IntegerPointsOnSegment(const point &P1, const point &P2){

point P = P1-P2;

P.x = abs(P.x); P.y = abs(P.y);

if(P.x == 0) return P.y;

if(P.y == 0) return P.x;

return (\_\_gcd(P.x,P.y));

}

Se asume que los vertices tienen coordenadas enteras. Sumar el valor de esta

funcion para todas las aristas para obtener el numero total de punto en el borde

del poligono.

\*/

// O(n log n)

// Entender que el convexhull te elimina los puntos repetidos :)

vector <Point> ConvexHull(vector <Point> Poly)

{

sort(Poly.begin(),Poly.end());

int nP = Poly.size(),k = 0;

Point H[ 2\*nP ];

for( int i = 0 ; i < nP ; ++i ){

while( k >= 2 && area( H [ k - 2 ] , H[ k - 1 ] , Poly[ i ] ) <= 0) --k;

H[ k++ ] = Poly[ i ];

}

for( int i = nP - 2 , t = k ; i >= 0 ; --i ){

while( k > t && area( H[ k - 2 ] , H[ k - 1 ] , Poly[ i ] ) <= 0) --k;

H[ k++ ] = Poly[ i ];

}

if( k == 0 )return vector <Point>();

return vector <Point> ( H , H + k - 1 );

}

//### DETERMINA SI P ESTA EN EL INTERIOR DEL POLIGONO CONVEXO A ########################

// O (log n)

bool isInConvex(vector <Point> &A, const Point &P)

{

int n = A.size(), lo = 1, hi = A.size() - 1;

if(area(A[0], A[1], P) <= 0) return 0;

if(area(A[n-1], A[0], P) <= 0) return 0;

while(hi - lo > 1)

{

int mid = (lo + hi) / 2;

if(area(A[0], A[mid], P) > 0) lo = mid;

else hi = mid;

}

return area(A[lo], A[hi], P) > 0;

}

// O(n)

Point norm(const Point &A, const Point &O)

{

Vector V = A - O;

V = V \* 10000000000.0 / V.mod();

return O + V;

}

bool isInConvex(vector <Point> &A, vector <Point> &B)

{

if(!isInConvex(A, B[0])) return 0;

else

{

int n = A.size(), p = 0;

for(int i=1; i<B.size(); i++)

{

while(!intersects(A[p], A[(p+1)%n], norm(B[i], B[0]), B[0])) p = (p+1)%n;

if(area(A[p], A[(p+1)%n], B[i]) <= 0) return 0;

}

return 1;

}

}

// INTERSECCION DE CIRCULOS

vector <Point> circleCircleIntersection(Point O1, double r1, Point O2, double r2)

{

vector <Point> X;

double d = dist(O1, O2);

if(d > r1 + r2 || d < max(r2, r1) - min(r2, r1)) return X;

else

{

double a = (r1\*r1 - r2\*r2 + d\*d) / (2.0\*d);

double b = d - a;

double c = sqrt(abs(r1\*r1 - a\*a));

Vector V = (O2-O1).unit();

Point H = O1 + V \* a;

X.push\_back(H + V.ort() \* c);

if(c > EPS) X.push\_back(H - V.ort() \* c);

}

return X;

}

// LINEA AB vs CIRCULO (O, r)

// 1. Mucha perdida de precision, reemplazar por resultados de formula.

// 2. Considerar line o segment

vector <Point> lineCircleIntersection(Point A, Point B, Point O, long double r)

{

vector <Point> X;

Point H1 = O + (B - A).ort() \* cross(O - A, B - A) / (B - A).mod2();

long double d2 = cross(O - A, B - A) \* cross(O - A, B - A) / (B - A).mod2();

if(d2 <= r\*r + EPS)

{

long double k = sqrt(abs(r \* r - d2));

Point P1 = H1 + (B - A) \* k / (B - A).mod();

Point P2 = H1 - (B - A) \* k / (B - A).mod();

if(between(A, B, P1)) X.push\_back(P1);

if(k > EPS && between(A, B, P2)) X.push\_back(P2);

}

return X;

}

/\*

// My own version

vector< Point > lineCircleIntersection( Point A , Point B , Point O , ld R ){

Vector V = ( B - A ).ort();

Point H = lineIntersection( A , B , O , O + V );

if( R < ( O - H ).dist() - EPS ) return vector< Point >();

if( abs( R - ( O - H ).dist() ) < EPS ) return vector< Point >( 1 , H );

vector< Point > ans;

Point P , Q;

Vector W = ( B - A ).unit();

ld d = sqrt( R\*R - ( O - H ).dist2() );

P = H + W\*d , Q = H - W\*d;

ans.pb( P ) , ans.pb( Q );

return ans;

}

\*/

//### PROBLEMAS BASICOS ###############################################################

void CircumscribedCircle()

{

int x1, y1, x2, y2, x3, y3;

scanf("%d %d %d %d %d %d", &x1, &y1, &x2, &y2, &x3, &y3);

Point A(x1, y1), B(x2, y2), C(x3, y3);

Point P1 = (A + B) / 2.0;

Point P2 = P1 + (B-A).ort();

Point P3 = (A + C) / 2.0;

Point P4 = P3 + (C-A).ort();

Point CC = lineIntersection(P1, P2, P3, P4);

double r = dist(A, CC);

printf("(%.6lf,%.6lf,%.6lf)\n", CC.x, CC.y, r);

}

void InscribedCircle()

{

int x1, y1, x2, y2, x3, y3;

scanf("%d %d %d %d %d %d", &x1, &y1, &x2, &y2, &x3, &y3);

Point A(x1, y1), B(x2, y2), C(x3, y3);

Point AX = A + (B-A).unit() + (C-A).unit();

Point BX = B + (A-B).unit() + (C-B).unit();

Point CC = lineIntersection(A, AX, B, BX);

double r = abs(area(A, B, CC) / dist(A, B));

printf("(%.6lf,%.6lf,%.6lf)\n", CC.x, CC.y, r);

}

vector <Point> TangentLineThroughPoint(Point P, Point C, long double r)

{

vector <Point> X;

long double h2 = (C - P).mod2();

if(h2 < r\*r) return X;

else

{

long double d = sqrt(h2 - r\*r);

long double m1 = (r\*(P.x - C.x) + d\*(P.y - C.y)) / h2;

long double n1 = (P.y - C.y - d\*m1) / r;

long double n2 = (d\*(P.x - C.x) + r\*(P.y - C.y)) / h2;

long double m2 = (P.x - C.x - d\*n2) / r;

X.push\_back(C + Point(m1, n1)\*r);

if(d != 0) X.push\_back(C + Point(m2, n2)\*r);

return X;

}

}

void TangentLineThroughPoint()

{

int xc, yc, r, xp, yp;

scanf("%d %d %d %d %d", &xc, &yc, &r, &xp, &yp);

Point C(xc, yc), P(xp, yp);

double hyp = dist(C, P);

if(hyp < r) printf("[]\n");

else

{

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

double m1 = (r\*(P.x - C.x) + d\*(P.y - C.y)) / (r\*r + d\*d);

double n1 = (P.y - C.y - d\*m1) / r;

double ang1 = 180 \* atan(-m1/n1) / PI + EPS;

if(ang1 < 0) ang1 += 180.0;

double n2 = (d\*(P.x - C.x) + r\*(P.y - C.y)) / (r\*r + d\*d);

double m2 = (P.x - C.x - d\*n2) / r;

double ang2 = 180 \* atan(-m2/n2) / PI + EPS;

if(ang2 < 0) ang2 += 180.0;

if(ang1 > ang2) swap(ang1, ang2);

if(d == 0) printf("[%.6lf]\n", ang1);

else printf("[%.6lf,%.6lf]\n", ang1, ang2);

}

}

void CircleThroughAPointAndTangentToALineWithRadius()

{

int xp, yp, x1, y1, x2, y2, r;

scanf("%d %d %d %d %d %d %d", &xp, &yp, &x1, &y1, &x2, &y2, &r);

Point P(xp, yp), A(x1, y1), B(x2, y2);

Vector V = (B - A).ort() \* r / (B - A).mod();

Point X[2];

int cnt = 0;

Point H1 = P + (B - A).ort() \* cross(P - A, B - A) / (B - A).mod2() + V;

double d1 = abs(r + cross(P - A, B - A) / (B - A).mod());

if(d1 - EPS <= r)

{

double k = sqrt(abs(r \* r - d1 \* d1));

X[cnt++] = Point(H1 + (B - A).unit() \* k);

if(k > EPS) X[cnt++] = Point(H1 - (B - A).unit() \* k);

}

Point H2 = P + (B - A).ort() \* cross(P - A, B - A) / (B - A).mod2() - V;

double d2 = abs(r - cross(P - A, B - A) / (B - A).mod());

if(d2 - EPS <= r)

{

double k = sqrt(abs(r \* r - d2 \* d2));

X[cnt++] = Point(H2 + (B - A).unit() \* k);

if(k > EPS) X[cnt++] = Point(H2 - (B - A).unit() \* k);

}

sort(X, X + cnt);

if(cnt == 0) printf("[]\n");

else if(cnt == 1) printf("[(%.6lf,%.6lf)]\n", X[0].x, X[0].y);

else if(cnt == 2) printf("[(%.6lf,%.6lf),(%.6lf,%.6lf)]\n", X[0].x, X[0].y, X[1].x, X[1].y);

}

void CircleTangentToTwoLinesWithRadius()

{

int x1, y1, x2, y2, x3, y3, x4, y4, r;

scanf("%d %d %d %d %d %d %d %d %d", &x1, &y1, &x2, &y2, &x3, &y3, &x4, &y4, &r);

Point A1(x1, y1), B1(x2, y2), A2(x3, y3), B2(x4, y4);

Vector V1 = (B1 - A1).ort() \* r / (B1 - A1).mod();

Vector V2 = (B2 - A2).ort() \* r / (B2 - A2).mod();

Point X[4];

X[0] = lineIntersection(A1 + V1, B1 + V1, A2 + V2, B2 + V2);

X[1] = lineIntersection(A1 + V1, B1 + V1, A2 - V2, B2 - V2);

X[2] = lineIntersection(A1 - V1, B1 - V1, A2 + V2, B2 + V2);

X[3] = lineIntersection(A1 - V1, B1 - V1, A2 - V2, B2 - V2);

sort(X, X + 4);

printf("[(%.6lf,%.6lf),(%.6lf,%.6lf),(%.6lf,%.6lf),(%.6lf,%.6lf)]\n", X[0].x, X[0].y, X[1].x, X[1].y, X[2].x, X[2].y, X[3].x, X[3].y);

}

void CircleTangentToTwoDisjointCirclesWithRadius()

{

int x1, y1, r1, x2, y2, r2, r;

scanf("%d %d %d %d %d %d %d", &x1, &y1, &r1, &x2, &y2, &r2, &r);

Point A(x1, y1), B(x2, y2);

r1 += r;

r2 += r;

double d = dist(A, B);

if(d > r1 + r2 || d < max(r1, r2) - min(r1, r2)) printf("[]\n");

else

{

double a = (r1\*r1 - r2\*r2 + d\*d) / (2.0\*d);

double b = d - a;

double c = sqrt(abs(r1\*r1 - a\*a));

Vector V = (B-A).unit();

Point H = A + V \* a;

Point P1 = H + V.ort() \* c;

Point P2 = H - V.ort() \* c;

if(P2 < P1) swap(P1, P2);

if(P1 == P2) printf("[(%.6lf,%.6lf)]\n", P1.x, P1.y);

else printf("[(%.6lf,%.6lf),(%.6lf,%.6lf)]\n", P1.x, P1.y, P2.x, P2.y);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*UTILITIES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*++

ll modulo( int num ){

( ( num %= mod ) += mod ) %= mod ;

return num ;

}

const ll mod = (int)( 1e+6 + 3 ) ;

const ld EPS=1e-6;

vector< int > pr , ex;

vector< int > getdivisors( ll x ){

vector< int > divisors( 1 , 1 );

for ( int i = 0 ; i < (int)pr.size() ; i++ ){

int m = (int)divisors.size();

for ( int j = 0 ; j < ex[ i ]; j++ ){

for ( int k = 0 ; k < m; k++ ) divisors.push\_back( divisors[ m\*j + k ]\*pr[ i ] );

}

}

return divisors;

}

void f( ll x ){

for ( ll i = 2 ; i \* i <= x ; i++ ){

if ( x % i == 0 ){

int cnt = 0;

while ( x % i == 0 ){

x /= i;

cnt++;

}

pr.push\_back( i );

ex.push\_back( cnt );

}

}

if ( x > 1 ){

pr.push\_back( x );

ex.push\_back( 1 );

}

}//generador de divisores de un numero x

\_\_builtin\_popcountll(n); //cantidad de unos de n en binario;

n long long;

\_\_builtin\_popcount(n); //cantidad de unos de n en binario; n int;

31 - \_\_builtin\_popclz(n); // posicion del mayor uno

(maxima potencia de 2 menor a n); n int;

63 - \_\_builtin\_popclz(n);//posicion del mayor uno

(maxima potencia de 2 menor a n); n long long;

\_\_builtin\_ctz(n);//cantidad de '0' de n en binario

\*\*\*\*\*VALORES DE CARACTERES\*\*

a en entero es 97

A........... 65

'.' ....... 46

'0' ....... 48

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DJISTRA\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

struct Node{

int dist,u;

Node(){}

Node(int \_dist,int \_u){

dist=\_dist;

u=\_u;

}

};

bool operator < (const Node a,const Node b){

return a.dist > b.dist;

}

vector<int> djistra(int source){

vector<int> d(998002, INF );

priority\_queue <Node> Q;

Q.push(Node(0,source));

d[source]=0;

while(!Q.empty()){

Node a=Q.top();

Q.pop();

for(int i=0;i<G[a.u].size();i++){

int v=G[a.u][i];

int w=C[a.u][i];

if(d[a.u] + w < d[v]){

d[v]=d[a.u] + w;

Q.push(Node(d[v],v));

}

}

}

return d;}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BFS 0-1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

const int INF=(1e9);

char M[1005][1005];

int d[1005][1005];

int dx[]={0,0,-1,1};

int dy[]={-1,1,0,0};

int n,m;

void bfs(int x,int y){

for(int i=1;i<=1000;i++){

for(int j=1;j<=1000;j++) d[i][j]=INF;

}

d[x][y]=0;

deque<pii> D;

D.push\_front(mp(x,y));

while(!D.empty()){

pii p = D.front();

D.pop\_front();

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

pii q = mp(p.fi + dx[i],p.se + dy[i]);

if(q.fi<1 || q.fi>n || q.se<1 || q.se>m) continue;

if(M[q.fi][q.se]=='X' &&

d[q.fi][q.se] > d[p.fi][p.se]){//EN CASO EL VALOR DEL EDGE ES 1

d[q.fi][q.se] = d[p.fi][p.se];

D.push\_front(q);}

else if(M[q.fi][q.se]=='.' &&

d[q.fi][q.se] > d[p.fi][p.se] + 1{//EN CASO EL VALOR DEL EDGE ES 0

d[q.fi][q.se] = d[p.fi][p.se] + 1;

D.push\_back(q);}

}

} }

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*VALORES CLASICOS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**DESVIACION ESTANDAR:**

int main(){

ll n;

while(cin>>n){

if(n==0) break;

ll sum=0;

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

ll val=2\*i - 1;

sum+=sqr(val-n);

}

double ans=sqrt( (double) sum / double(n-1) );

printf("%.6f\n",ans);

}

return 0;

}

**DP con mask:**

int memo[12][4098];

int memo1[13];

int n;

int fact(int num){

if(num==0) return 1;

if(memo1[num] > 0) return memo1[num];

int &ans=memo1[num]=1;

ans=num\*fact(num-1);

return ans;

}

int dp(int num,int mask){

if(num==(n-1)){

if(mask & (1<<0) ) return 0;

else return 1;

}

if(memo[num][mask] > 0) return memo[num][mask];

int &ans=memo[num][mask]=0;

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

if( mask & ( 1<<(n-i-1) ) ){

if(i==num) continue;

ans+=dp(num+1,mask ^ (1<<(n-i-1)));

}

}

return ans;

}

int main(){

int t;cin>>t;

while(t>0){

t--;

cin>>n;

int divi=fact(n);

int num=dp(0, (1<<n) - 1);

cout<<num<<"/"<<divi<<'\n';

}

return 0;

}

**BFS MIKEMON:**

int M[1001][1001];

char N[1001][1001];

int dx[]={-1,1,0,0};

int dy[]={0,0,-1,1};

bool vis[1001][1002];

int n,m;

void bfs(int x,int y){

vis[x][y]=1;

queue<pii> Q;

Q.push(mp(x,y));

while(!Q.empty()){

pii p=Q.front();

int px=p.fi,py=p.se;

Q.pop();

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

int xx=px+dx[i];

int yy=py+dy[i];

if( xx < 1 || xx > n || yy < 1 || yy > m) continue;

if(N[xx][yy]=='T') continue;

if(!vis[xx][yy]){

M[xx][yy]=M[px][py]+1;

vis[xx][yy]=1;

Q.push(mp(xx,yy));

}

}

}

}

int main(){

cin>>n>>m;

memset(M,-1,sizeof(M));

int x,y;

int xx,yy;

vector< pair < pii,int > > mik;

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

for(int j=1;j<=m;j++){

cin>>N[i][j];

if(N[i][j]=='S'){

x=i;y=j;

}

if(N[i][j]=='E'){

xx=i;yy=j;

}

if(N[i][j]>='1' && N[i][j]<='9') mik.pb(mp(mp(i,j) , N[i][j]-'0'));

}

}

bfs(xx,yy);

int res=M[x][y];

int ans=0;

for(int i=0;i<sz(mik);i++){

if(M[mik[i].fi.fi][mik[i].fi.se]==(-1)) continue;

if(M[mik[i].fi.fi][mik[i].fi.se] <= res) ans+=mik[i].se;

}

cout<<ans<<endl;

return 0;

}

**DP TSP:**

int M[12][102];

int memo[12][102];

int n,m;

int dp(int i,int j){

if(memo[i][j]) return memo[i][j];

if(j==m) {memo[i][j]=M[i][j];return M[i][j];}

int &ans=memo[i][j]=0;

ans+=M[i][j];

ans+=min( min( dp(i-1 + n\*(i==1),j+1) , dp(i , j+1) ) , dp(i+1 - n\*(i==n),j+1) );

return ans;

}

void rec(int val,int pos,int col){

cout<<pos;

val-=M[pos][col];

if(col==m) return;

cout<<" ";

if(pos==1){

if(memo[pos][col+1]==val){rec(val,pos,col+1);}

else if(memo[pos+1][col+1]==val){rec(val,pos+1,col+1);}

else rec(val,n,col+1);

}

else if(pos==n){

if(memo[1][col+1]==val){rec(val,1,col+1);}

else if(memo[pos-1][col+1]==val){rec(val,pos-1,col+1);}

else rec(val,pos,col+1);

}

else{

if(memo[pos-1][col+1]==val) {rec(val,pos-1,col+1);}

else if(memo[pos][col+1]==val){rec(val,pos,col+1);}

else {rec(val,pos+1,col+1);}

}

}

int main(){

//fast\_io();

while(cin>>n>>m){

memset(M,0,sizeof(M));

memset(memo,0,sizeof(memo));

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

for(int j=1;j<=m;j++){

cin>>M[i][j];

}

}

int mini=(1<<30);

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

mini=min(mini,dp(i,1));

}

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

if(memo[i][1]==mini) {rec(mini,i,1);break;}

}

cout<<endl;

cout<<mini<<endl;

}

return 0;

}

**TWO POINTER-STL:**

set<char> S;

map<char,int> M;

bool cumple(){

map<char,int> :: iterator it2;

for(it2=M.begin();it2!=M.end();it2++){

if((\*it2).se==0) return false;

}

return true;

}

int main(){

//fast\_io();

int n;cin>>n;

string s;cin>>s;

int val=0;

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

S.insert(s[i]);

}

set<char> :: iterator it;

for(it=S.begin();it!=S.end();it++){

M[(\*it)]=0;

}

int j=-1;

int mini=1000000;

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

if(cumple()){

mini=min(mini,j-i+1);

M[s[i]]--;

continue;

}

while( (j+1) < n ){

j++;

M[s[j]]++;

if(cumple()){

mini=min(mini,j-i+1);

break;

}

}

M[s[i]]--;

}

cout<<mini<<endl;

return 0;

}

**TWO POINTER-BINARY SEARCH:**

vector<ll> city;

vector<ll> tower;

bool f(ll r){

int j=-1;

for(int i=0;i<sz(tower);i++){

while((j+1)<sz(city)){

if( city[j+1] >= tower[i]-r && city[j+1] <= tower[i] + r ) j++;

else break;

}

}

if(j+1==sz(city)) return true;

else return false;

}

int main(){

fast\_io();

int n,m;cin>>n>>m;

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

ll a;cin>>a;city.pb(a);

}

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

ll a;cin>>a;tower.pb(a);

}

sort(all(city));

sort(all(tower));

if(f(0)) {cout<<0<<endl;return 0;}

ll lo=0.0,hi=(2e9);

while((hi-lo) > 1){

ll mi=(lo+hi)/2;

if(f(mi)) hi=mi;

else lo=mi;

}

cout<<hi<<endl;

return 0;

}

**TWO POINTER CLASSIC:**

map<int,int> M;

bool cumple(){

map<int,int> :: iterator it;

if(M.size() > 2) return false;

if(M.size() < 2) return true;

for(it=M.begin();it!=M.end();it++){

}

}

int main(){

//fast\_io();

int n;cin>>n;

vi v(n);

for(int i=0;i<n;i++) cin>>v[i];

int maxi=0;

int j=-1;

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

while((j+1) < n){

if(cumple()) j++;

else{

maxi=max(maxi,j-i);

}

}

}

return 0;

}

**TERNARY SEARCH:**

double f(double val){

//funcion que rige la cuadratica;

}

int main(){

//fast\_io();

double hi,lo;

for(int i=0;i<100;i++){

int mi1=hi - (hi-lo)/3.0;

int mi2=lo + (hi-lo)/3.0;

if(f(mi2) < f(mi1)) lo=mi2;

else hi=mi1;

}

printf("%.10f\n",(hi+lo)/2.0);

return 0;

}

**UNION FIND (CON RANK):**

int pa[1002]; //todos se inicializa en pa[i] = i;

int ranked[1002];

int Find(int i){

if(pa[i]==i) return i;

return find(pa[i]);

}

void Union(int x,int y){

int xset=find(x);

int yset=find(y);

if(ranked[xset]<ranked[yset]){

pa[xset]=yset;

}

else if(ranked[xset]>ranked[yset]){

pa[yset]=xset;

}

else{

pa[yset]=xset;

ranked[xset]++;

}

}

**UNION FIND (ENEMIES ANS FRIENDS):** //WAR UVA 10158

const int N=(1e4);

int pa[2\*N+5];

int ranked[2\*N+5];

int Find(int i){

if(pa[i]==i) return i;

return Find(pa[i]);

}

void Union(int x,int y){

int xset=Find(x);

int yset=Find(y);

if(ranked[xset]<ranked[yset]){

pa[xset]=yset;

}

else if(ranked[xset]>ranked[yset]){

pa[yset]=xset;

}

else{

pa[yset]=xset;

ranked[xset]++;

}

}

// Para setear amigos Union(a,b);Union(a+n,b+n);

// Para setear enemigos Union(a+n,b);Union(a,b+n);

**NUMBER THEORY**

**EXTENDIDO DE EUCLIDES:**

//Se utiliza para resolver ecuaciones del tipo ax+by=gcd(a,b);

typedef pair<ll,pair<ll,ll> > tup;

tup extGcd(ll a,ll b){

if(b==0) return mp(a,mp(1,0));

tup ret = extGcd(b,a%b);

return mp(ret.fi , mp(ret.se.se, ret.se.fi - (a/b)\* ret.se.se));

}

/\*Si deseas hallar para una ecuacion diofantica en general se debe cumplir:

ax+by=k , k es multiplo de gcd(a,b)\*/

**INVERSO MODULAR:**

//Debe cumplirse gcd(a,n)=1;

SI n ES PRIMO entonces:

const ll MOD=(1e9 + 7);

ll pot(ll a,ll b){

if(b==0) return 1;

if(b==1) return a;

ll ans=1;

if(b&1) ans\*=a;

ans\*=pot(a,b/2);

ans%=MOD;

ans\*=pot(a,b/2);

ans%=MOD;

return ans;

}

ll inv(ll a){

return pot(a,MOD-2)%MOD;

}

SI n NO ES PRIMO entonces:

ll inv(ll a,ll n){

tup t= extGcd(a,n);

ll inver=((t.se.fi%n) + n)%n;

return inver;

}

Se utiliza la función de Extendido de Euclides

**CHINESSE THEOREM REMAINDER**

x = a1 (mod m1)

x = a1 (mod m2)

.

.

.

x = ak (mod mk)

m1, m2, m3 ,...., mk son PESI

entonces:

x = a1m1y1 + a2m2y2 + .... + akmkyk

donde mi = inverso modular de yi mod ni

ll chinese(vector<ll>&rem , vector<ll>&mod){

int k=sz(mod);

ll n=1;

for(int i=0;i<k;i++) n\*=mod[i];

ll x=0;

for(int i=0;i<k;i++){

ll m=n/mod[i];

ll y=inv(m,mod[i]);

y%=n;

x+=(rem[i] \* ( ( m\*y) % n))%n;

x%=n;

}

return x;

}

**TEOREMA DE LUCAS:**

ll C[105][105];

int Lucas(int n, int r, int p){

if (r==0) return 1;

int ni = n%p, ri = r%p;

return (Lucas(n/p, r/p, p) \* C[ni][ri] % p) % p;

}

for(int i=0;i<=50;i++) C[i][0]=1; // inicializamos

for(int i=1;i<=50;i++){

for(int j=i;j<=50;j++){

if(i==j) C[i][j]=1;

else C[j][i]=C[j-1][i-1]+C[j-1][i];

}

}

**HACKERRANK PROBLEM – TEOREMA DEL CHINO Y LUCAS**

#include <bits/stdc++.h>

typedef long long ll;

using namespace std;

ll mod;

vector<ll> pr;

vector<ll> res;

ll pot(ll a,ll b,ll c){

if(b==0) return 1;

if(b==1) return a;

ll ans=1;

if(b&1) ans\*=a;

ans\*=pot(a,b/2,c);

ans%=c;

ans\*=pot(a,b/2,c);

ans%=c;

return ans;

}

ll inv(ll a,ll b){

return pot(a,b-2,b)%b;

}

void f(ll x){

for(ll i=2;i\*i<=x;i++){

if(x%i==0){

pr.push\_back(i);

x/=i;

}

}

if(x>1) pr.push\_back(x);

}

ll chino(){

ll x=0;

for(int i=0;i<pr.size();i++){

ll m=mod/pr[i];

ll y=inv(m,pr[i]);

y%=mod;

x+=(res[i] \* ( ( m\*y) % mod))%mod;

x%=mod;

}

return x;

}

ll C[105][105];

int Lucas(int n, int r, int p)

{

if (r==0)

return 1;

int ni = n%p, ri = r%p;

return (Lucas(n/p, r/p, p) \* // Last digits of n and r

C[ni][ri] % p) % p; // Remaining digits

}

int main() {

int t;cin>>t;

ll a,b;

for(int i=0;i<=50;i++) C[i][0]=1;

for(int i=1;i<=50;i++){

for(int j=i;j<=50;j++){

if(i==j) C[i][j]=1;

else C[j][i]=C[j-1][i-1]+C[j-1][i];

}

}

while(t--){

cin>>a>>b>>mod;

pr.clear();

res.clear();

f(mod);

for(int i=0;i<pr.size();i++){

res.push\_back(Lucas(a,b,pr[i]));

}

ll ans=chino();

cout<<ans<<endl;

}

return 0;

}

**EXPONENCIACION RAPIDA EN COMPLEJOS:**

typedef long long ll;

using namespace std;

ll MOD;

pair<ll,ll> mult(pair<ll,ll> a,pair<ll,ll> b){

pair<ll,ll> ans=make\_pair(0,0);

ans.first+=(a.first\*b.first);

ans.first%=MOD;

ans.second+=(a.second\*b.first);

ans.second%=MOD;

ans.second+=(a.first\*b.second);

ans.second%=MOD;

ans.first-=(a.second\*b.second);

ans.first%=MOD;

ans.first+=MOD;

ans.first%=MOD;

return ans;

}

pair<ll,ll> pot(pair<ll,ll> a,ll b){

if(b==0) return make\_pair(1,0);

if(b==1) return a;

pair<ll,ll> ans=make\_pair(1,0);

if(b&1) ans=a;

pair<ll,ll> val = pot(a,b/2);

ans=mult(ans,val);

ans=mult(ans,val);

return ans;

}

int main() {

int q;cin>>q;

ll a,b,k;

while(q--){

cin>>a>>b>>k>>MOD;

pair<ll,ll> p = pot(make\_pair(a,b),k);

cout<<p.first<<" "<<p.second<<endl;

}

return 0;

}

**NUMERO DE FIBONACCI EFICIENTE CON DISTINTAS INICIALES**

using namespace std;

#define long long ll

const ll MOD = 1000000007;

map<ll, ll> M;

ll f(ll n) {

if (M.count(n)) return M[n];

long k=n/2;

if (n%2==0)

return M[n] = ((f(k)\*f(k))%MOD + (f(k-1)\*f(k-1))%MOD) %MOD;

else

return M[n] = ((f(k)\*f(k+1))%MOD + (f(k-1)\*f(k))%MOD) % MOD;

}

int main(){

ll n;

int t;cin>>t;

M[0]=M[1]=1;

ll a,b;

while (t--){

cin>>a>>b>>n;

if(n==0) cout<<a<<endl;

else if(n==1) cout<<b<<endl;

else cout<<( (a\*f(n-2))%MOD + (b\*f(n-1))%MOD )%MOD<<endl;

}

**MOBIUS:**

MOBIUS, es una funcion multiplicativa que esta definida de la siguiente forma:

u(n) = 1 , si n es LC y cantidad de factores primos par

u(n) = -1 , si n es LC y cantidad de factores primos impar

u(n) = 0 , si n es no LC

LC: Libre de Cuadrados

para hallar el mobius de manera eficiente utilizamos criba

const int UP=(1e4);

int fact[UP + 5];

int mu[UP + 5];

ll D[UP + 5];

void criba(){

for(int i=0;i<=UP;i++) fact[i]=-1;

for(int i=2;i\*i<=UP;i++){

if(fact[i]==-1){

for(int j=i\*i;j<=UP;j+=i){

if(fact[j]==-1) fact[j]=i;

}

}

}

}

void mobius(){

mu[1]=1;

for(int i=2;i<=UP;i++){

if(fact[i]==-1) mu[i]=-1;

else{

int nx=i/fact[i];

if(nx % fact[i]==0) mu[i]=0;

else mu[i]=-mu[nx];

}

}

}

int main(){

fast\_io();

int n;

criba();

mobius();

while (cin>>n){

memset( D,0,sizeof(D));

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

int x; cin>>x;

for ( int j = 1; j <= x ; j++ ){

if ( x % j == 0 ) D[j]++;

}

}

long long ans = 0;

for ( int i = 1; i < UP;i++ ){

long long val = D[i];

val = (val)\*(val-1)\*(val-2)\*(val-3)/24; ans += val\*mu[i];

}

cout<<ans<<endl;

}

return 0 ;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RANGE MINIMUN QUERY\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Usando Sparse Table

const int N=(1e5);

const int MAXN=(60);

using namespace std;

int M[N+5][MAXN+5];

int n;

int f(int x,int y){

return log2(y-x+1);

}

int main() {

cin>>n;

vector<int> A(n);

for(int i=0;i<n;i++) cin>>A[i];

//initialize M for the intervals with length 1

for (int i = 0; i < n; i++) M[i][0] = i;

//compute values from smaller to bigger intervals

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

for (int i = 0; i + (1 << j) - 1 < n; i++){

if (A[M[i][j - 1]] < A[M[i + (1 << (j - 1))][j - 1]])

M[i][j] = M[i][j - 1];

else M[i][j] = M[i + (1 << (j - 1))][j - 1];

}

}

int q;cin>>q;

int a,b;

while(q--){

cin>>a>>b;

int pos = f(a,b);

cout<<min(A[M[a][pos]] ,

A[M[b - (1<<pos) + 1][pos]])<<endl;

}

return 0;

}