组合数C(n,m)：

O(n)：

ll F[N], Finv[N], inv[N];//F是阶乘，Finv是逆元的阶乘

ll n,m;

void init(){

inv[1] = 1;

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

inv[i] = (mod - mod / i) \* 1ll \* inv[mod % i] % mod;

}

F[0] = Finv[0] = 1;

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

F[i] = F[i-1] \* 1ll \* i % mod;

Finv[i] = Finv[i-1] \* 1ll\* inv[i] % mod;

}

}

ll C(ll n, ll m){ //C(n, m)

if(m < 0 || m > n) return 0;

return F[n] \* 1ll \* Finv[n - m] % mod \* Finv[m] % mod;

}

一行快速幂：

ll fast(ll a,ll b){ll ans=1;for(;b;b>>=1,a=mul(a,a))if(b&1)ans=mul(ans,a);return ans;}

快速乘：

ll mod\_mul(ll a, ll b, ll mod) {

ll ret = 0;

while(b) {

if(b & 1) ret = (ret + a) % mod;

a = (a + a) % mod;

b >>= 1;

}

return ret;

}

矩阵快速幂：

**struct** Mt {**ll** m[3][3];};

Mt mul(Mt a, Mt b) {

Mt ret;

**for** (**int** i = 0; i < 3; i++)

**for** (**int** j = 0; j < 3; j++) {

ret.m[i][j] = 0;

**for** (**int** k = 0; k < 3; k++) {

ret.m[i][j] = (ret.m[i][j] + a.m[i][k] \* b.m[k][j] % mod) % mod;

}

}

**return** ret;

}

Mt qpow(Mt a, **ll** n) {

Mt ret;

memset(ret.m, 0, **sizeof**(ret.m));

ret.m[0][0] = ret.m[1][1] = ret.m[2][2] = 1;

Mt tmp = a;

**while** (n) {

**if** (n & 1)ret = mul(ret, tmp);

tmp = mul(tmp, tmp);

n >>= 1;

}

**return** ret;

}

素数筛(O(n)):

int pri[N];

int cnt;

bool v[N];

void init()

{

cnt = 0;//素数筛

v[1]=1;

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

{

if(!v[i])

pri[cnt++] = i;

for(int j = 0; j < cnt && pri[j] <= N/i; j++)

{

v[i\*pri[j]] = 1;

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

}

}

}

莫比乌斯函数表：

bool vis[N + 10];

int pri[N + 10];

int mu[N + 10];

void mus() {

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

mu[1] = 1;

int tot = 0;

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

if (!vis[i]) {

pri[tot++] = i;

mu[i] = -1;

}

for (int j = 0; j < tot && i \* pri[j] < N; j++) {

vis[i \* pri[j]] = 1;

if (i % pri[j] == 0) {

mu[i \* pri[j]] = 0;

break;

}

else mu[i \* pri[j]] = -mu[i];

}

}

}

欧拉函数表：

const int N = 1e6+10 ;

int phi[N];//函数值

int pri[N];

int tot;//tot计数，表示pri[N]中有多少质数

void Euler(){

phi[1] = 1;

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

if(!phi[i]){

phi[i] = i-1;

pri[tot ++] = i;

}

for(int j = 0; j < tot && 1ll\*i\*pri[j] < N; j ++){

if(i % pri[j]) phi[i \* pri[j]] = phi[i] \* (pri[j]-1);//互质

else{

phi[i \* pri[j] ] = phi[i] \* pri[j];//含有这个因子

break;

}

}

}

}

int main(){

Euler();

}

欧拉函数：

int phi(int n) {

int ans = n;

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

if (n % i == 0) {

ans -= ans / i;

while (n % i == 0) n /= i;

}

}

if (n > 1) ans -= ans / n;

return ans;

}

特殊性质：

1、若m,n互质，φ(mn)=φ(m)φ(n)

2、n=∑ϕ(d) （d|n）

3、对于给定的一个素数p，φ(p)=p -1。则对于正整数 n = p^k ，φ(n) = p^k - p^(k -1)

4、ϕ(n)=n\*(1-1/p1)\*(1-1/p2)…(1-1/pk).

递推性质：

若(n % p == 0) 则有：φ(n)=φ(n / p) \* p；

若(n % p != 0)  则有：φ(n)=φ(n / p) \* (p-1)。

一般化：

若(n % a == 0 && (n / a) % a == 0) 则有：φ(n)=φ(n / a) \* a；

若(n % a == 0 && (n / a) % a != 0)  则有：φ(n)=φ(n / a) \* (a - 1)。

扩展欧几里得：

ll exgcd(ll a, ll b, ll &x, ll &y)*//ax+by=gcd(a,b)*的整数解

{

ll d;

//if (a == 0 && b == 0) return -1;

if (b == 0)

{

x = 1,y = 0;

return a;

}

d = exgcd(b, a%b, y, x);

y -= a / b \* x;

return d;

}

ll inv(ll a,ll p)

{

ll d,x,y;

d = exgcd(a,p,x,y);

return d==1?(x%p+p)%p:-1;

}

简单应用：ax + by + c = 0在[x1, x2], [y1, y2]区间内有多少组解？

#include"bits/stdc++.h"

using namespace std;

ll exgcd(ll a, ll b, ll &x, ll &y)//ax+by=gcd(a,b)的整数解

{

ll d;

//if (a == 0 && b == 0) return -1;

if (b == 0)

{

x = 1,y = 0;

return a;

}

d = exgcd(b, a%b, y, x);

y -= a / b \* x;

return d;

}

ll gcd(ll x,ll y){

return y==0?x:gcd(y,x%y);

}

int main()

{

ll a,b,c,x1,x2,y1,y2;

cl(a),cl(b),cl(c),cl(x1),cl(x2),cl(y1),cl(y2);//输入a,b,-c,x1,x2,y1,y2.

ll ans;

c=-c;

ll d=gcd(a,b);

if(!a&&!b){

if(!c) pi((x2-x1+1)\*(y2-y1+1));

else puts("0");

}

else if(!a){

if(c%b==0&&y1<=c/b&&c/b<=y2) pi(x2-x1+1);

else puts("0");

}

else if(!b){

if(c%a==0&&x1<=c/a&&c/a<=x2) pi(y2-y1+1);

else puts("0");

}

else

{

if(c%d!=0) ans=0;

else

{

ll l1,r1,l2,r2;

a/=d,b/=d,c/=d;

ll x,y;

exgcd(a,b,x,y);

x\*=c,y\*=c;

l1=(x1-x)/b,r1=((x2-x)/b);

l2=(y-y2)/a,r2=((y-y1)/a);

if(l1>r2||r1<l2) ans=0;

else ans=min(r1,r2)-max(l1,l2)+1;

}

pl(ans);

}

return 0;

}

Miller\_Rabin:

ll mod\_mul(ll a, ll b, ll mod) {

ll ret = 0;

while(b) {

if(b & 1) ret = (ret + a) % mod;

a = (a + a) % mod;

b >>= 1;

}

return ret;

}

ll mod\_exp(ll a, ll b, ll mod) {

ll ret = 1;

while(b) {

if(b & 1) ret = mod\_mul(ret, a, mod);

a = mod\_mul(a, a, mod);

b >>= 1;

}

return ret;

}

bool check(ll a, ll n) {

ll x = n - 1, y;

int t = 0;

while((x & 1) == 0) {

x >>= 1;

t++;

}

x = mod\_exp(a, x, n);

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

y = mod\_exp(x, 2, n);

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

x = y;

}

if(y != 1) return true;

return false;

}

素性测试**:**

bool Miller\_Rabin(ll n, int times = 10) {

if(n == 2) return 1;

if(n == 1 || !(n & 1)) return 0;

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

ll a = rand() % (n - 1) + 1;

if(check(a, n)) return 0;

}

return 1;

}

大整数分解:

ll Pollard\_rho(ll n, int c) {

ll i = 1, k = 2, x, y, d;

y = x = rand() % n; //随机初始化一个基数（p1）

while(true) {

i++;

x = (mod\_mul(x, x, n) + c) % n; //玄学递推

d = \_\_gcd(y - x, n);

if(d > 1 && d < n) return d; //判断

if(y == x) break; //y为x的备份，相等则说明遇到圈，退出

if(i == k) {

y = x;

k <<= 1; //更新y，判圈算法应用

}

}

return n;

}

void factorize(ll n, int c = 107) { //n为待分解数，c为随机常数

if(n == 1) return;

if(Miller\_Rabin(n)) { //n为质数

factor[cnt++] = n; //保存

return;

}

ll p = n;

while(p >= n) p = Pollard\_rho(p, c--); //当未分解成功时，换个c带入算法

factorize(p, c);

factorize(n / p, c); //递归操作

}

**POJ 2429:** 已知**x,y**的**gcd**和**lcm**，求使**x+y**的值最小的**x**，**y**值。（**0<x,y<=1e18)**

#include"bits/stdc++.h"

typedef long long ll;

ll pr[100],f[100],t,ans;

ll mult(ll a,ll b,ll n)// a\*b % n

{

ll s=0;

while(b){

if(b&1)s=(s+a)%n;

a<<=1;b>>=1;

if(a>n)a=a%n;

}

return s;

}

ll miller\_rabin(ll n,ll i)//素数测定，错误机率为1/4

{

ll j,k,b=0,a=1;

j=(n-1)/(1<<i);

while(b%n==0)b=abs(rand())%n;

while(j){

if(j&1)a=mult(a,b,n);

b=mult(b,b,n);j>>=1;

}

for(k=1;k<=i;k++){

if(k<i-1&&mult(a,a,n)==1&&a!=1&&a!=n-1)return 0;

a=mult(a,a,n);

}

if(a==1)return 1;

return 0;

}

ll pri(ll n)

{//调用miller\_rabin素数测试10次,正确概率为1 - (1/4)^10,几乎为1

ll m,k,i,s=1;

if(n==1) return 0;

for(k=0,m=n-1;m%2==0;m/=2,k++);

for(i=0,s=1;i<10;i++)s\*=miller\_rabin(n,k);

return s;

}

ll gcd(ll a,ll b)//求最大公约数

{

return b?gcd(b,a%b):a;

}

ll pollard\_rho(ll n)////pollard\_rho随机算法求n的一个约数

{

ll i=1,a,b,p,t=2,c=0;

a=b=abs(rand())%(n-1)+1;

while(c==0||c==2)c=abs(rand())%(n-1)+1;

do{

i++;

p=gcd(n+b-a,n);

if(p>1&&p<n)return p;

if(i==t){

b=a;t<<=1;

}

a=(mult(a,a,n)+n-c)%n;

}while(a!=b);

return n;

}

void add(ll n)//将素因子添加到素因子表中

{

ll i;

for(i=0;i<t;i++)

if(pr[i]==n)break;

if(i<t)f[i]++;//第i个素因子个数加1

else{//如果之前没有添加

pr[t]=n;//添加到表尾

f[t++]=1;//设素因子个数初始化为1

}

}

void find(ll n)

{

ll p;

if(n<2)return;

if(pri(n))add(n);//如果n为素数，添加到素因子表中

else{//如果n不为素数，继续拆分n

p=pollard\_rho(n);

find(p);//尝试拆分n的一个约数p

find(n/p);//尝试拆分n的一个约数n/p

}

}

void findx(ll i,ll x,ll q)

{

if(i==t)return;

if(x>ans&&x<=q) ans=x;//如果约数x比ans更接近q，则更新ans

findx(i+1,x,q);

x\*=f[i];

if(x>ans&&x<=q) ans=x;//如果约数x比ans更接近q，则更新ans

findx(i+1,x,q);

}

int main()

{

ll i,j,k,m,n,q;

srand((unsigned)time(NULL));

while(~scanf("%I64d%I64d",&m,&n)){

t=0;n/=m;

find(n);//分解质因数

for(i=0;i<t;i++){//将相同素因子作为一个整体处理

for(j=1,k=pr[i];j<f[i];j++)k\*=pr[i];

f[i]=k;

}

q=(ll)sqrt(n\*1.0);

ans=1;

findx(0,1,q);//枚举n的因子,找出小于q且最接近q的因子

printf("%I64d %I64d\n",ans\*m,n/ans\*m);

}

return 0;

}

#include**<cstdio>** *//中国剩余定理（不互素版）*

#include**<algorithm>**

using namespacestd;

**typedef long long** ll;

**typedef** pair<ll, ll> pll;

pll CRT(ll A[], ll B[], ll M[], **int** n) {*//求解A[i]x = B[i] (mod M[i]),总共n个线性方程组*

ll x = 0, m = 1;

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

ll a = A[i] \* m, b = B[i] - A[i]\*x, d = gcd(M[i], a);

**if**(b % d != 0) **return** pll(0, -1);*//答案不存在，返回-1*

ll t = b/d \* inv(a/d, M[i]/d)%(M[i]/d);

x = x + m\*t;

m \*= M[i]/d;

}

x = (x % m + m ) % m;

**return** pll(x, m);*//返回的x就是答案，m是最后的lcm值*

}

*//n个方程：x=a[i](mod m[i]) (0<=i<n) //中国剩余定理*

ll china(**int** n, ll \*a, ll \*m){

ll M = 1, ret = 0;

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

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

ll w = M / m[i];

ret = (ret + w \* inv(w, m[i]) \* a[i]) % M;

}

**return** (ret + M) % M;

}

费马小定理：

当a,p互质，且p为素数的时候 a^(p-1)=1(mod p)

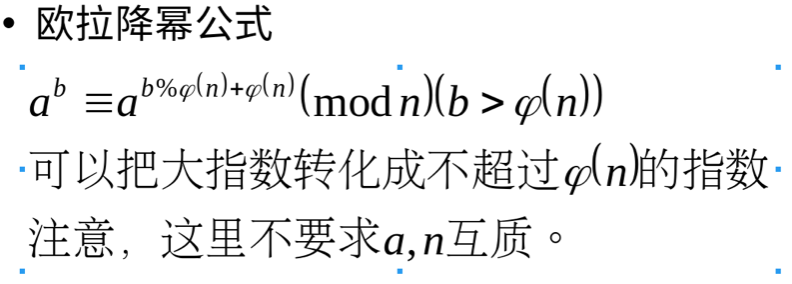
卢卡斯定理：大组合数

C(n, m) % p  =  C(n / p, m / p) \* C(n%p, m%p) % p

LL Lucas(LL n, LL m, int p){

return m ? Lucas(n/p, m/p, p) \* comb(n%p, m%p, p) % p : 1;

}

A^B mod C:

(1<=A,C<=1000000000,1<=B<=10^1000000).

#include<iostream>

#include<string>

#include<vector>

#include<algorithm>

#include<queue>

#include<cstdio>

#include<cstring>

#include<cmath>

#include<map>

using namespace std;

typedef long long ll;

const int INF\_INT = 0x3f3f3f3f;

const ll INF\_LL = 1e18;

const int MOD=1e9 + 7;

const int N = 100005;

**//**计算欧拉函数**O(sqrt(n))**

int Phi(int x)

{

int i,re=x;

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

if(x%i==0)

{

re/=i;re\*=i-1;

while(x%i==0)

x/=i;

}

if(x^1) re/=x,re\*=x-1;

return re;

}

ll qpow(ll x,ll y,ll p)

{

ll re=1;

while(y)

{

if(y&1) (re\*=x)%=p;

(x\*=x)%=p; y>>=1;

}

return re;

}

int cal(int p)

{

if(p==1) return 0;

int phi\_p = Phi(p);

return qpow(2,cal(phi\_p)+phi\_p,p);

}

int T,n,a,c;

string b;

int main()

{

ios\_base::sync\_with\_stdio(false);

while(cin>>a>>b>>c)

{

int phi\_c = Phi(c);

ll sum = 0;

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

{

sum = (sum\*10+b[i]-'0')%(phi\_c);

}

cout <<qpow(a,sum+phi\_c,c)<<endl;

}

}

Polya定理：

给出m种颜色和n个珠子,将珠子染色后穿成一个环,旋转和翻转相同的视为同构,求方案数:

ll gcd(ll x,ll y) {**return** y==0?x:gcd(y,x%y);}

ll qpow(ll x,ll n)

{

ll ans=1;

while(n){

**if**(n&1) ans=ans\*x;

x=x\*x;

n>>=1;

}

**return** ans;

}

**int** main()

{

ll n,m;

**while**(scanf(**"%lld%lld"**,&m,&n)==2&&n||m)

{

**if**(!n) puts(**"0"**);

**else**

{

ll ans=0;

**for**(ll i=1;i<=n;i++) ans+=qpow(m,gcd(n,i));

**if**(n&1) ans+=qpow(m,n/2+1)\*n;

**else** ans+=qpow(m,n/2+1)\*(n/2)+qpow(m,n/2)\*(n/2);

pl(ans/2/n);

}

}

**return** 0;

}

威尔逊定理**: (p-1)! mod p=-1 (p**是素数**)**

证明：

考虑一个长度为*p*的环，将环上*p*个点顺次连成一条折线共有*(p-1)!*种，其中*p-1*种全同的具有循环不变性，其他每个都属于大小为*p*的同余类故*p|((p-1)!-(p-1))*即*(p-1)!=-1(mod p)*

离散对数：解决同余方程 **A^x = B (mod C)**的最小整数解

**typedef long long** ll;

#define mod 99991

**struct** node{ll v,num,f;}hash[mod+10];

ll A,B,C;

ll gcd(ll a,ll b) {**return** !b?a:gcd(b,a%b);}

**void** insert(ll v,ll x) *//*哈希表插入

{

ll t=v%mod;

**while**(hash[t].f&&hash[t].v!=v) {t++; **if**(t>mod) t-=mod;}

**if**(!hash[t].f) {hash[t].f=1;hash[t].num=x;hash[t].v=v;}

}

ll find(ll v) *//*哈希表查询

{

ll t=v%mod;

**while**(hash[t].f&&hash[t].v!=v) {t++; **if**(t>mod) t-=mod;}

**if**(!hash[t].f) **return** -1;

**else return** hash[t].num;

}

**void** exgcd(ll a,ll b,ll &x,ll &y)

{

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

exgcd(b,a%b,x,y);

ll t=x;x=y;y=t-a/b\*y;

}

ll Shank()

{

ll ret=1;

**for**(ll i=0;i<=50;i++) {**if**(ret==B)**return** i;ret=ret\*A%C;}

*//*判断在*0~50*内是否有解

ll tmp,ans(1),cnt(0);

**while**((tmp=gcd(A,C))!=1)*//*使得*gcd(A,C)=1*为止

{

**if**(B%tmp) **return** -1;*//*根据裴蜀定理，如果*gcd(A,C)|B*不成立，则无解

C/=tmp; B/=tmp;

ans=ans\*(A/tmp)%C;

cnt++; *//*记录已经乘了*cnt*个*A*，以及这些*A*与*C*消掉公因子后的结果*ans*

}

ll m=(ll)ceil(sqrt(C\*1.0)),t(1);

**for**(ll i=0;i<m;i++) {insert(t,i);t=t\*A%C;}

**for**(ll i=0;i<m;i++)

{

ll x,y;

exgcd(ans,C,x,y);

ll val=x\*B%C; *//*求*A^(m\*i)\*x=B(mod C)*的解*x*

val=(val%C+C)%C; *//x*的最小整数解

ll j=find(val); *//*找到*val*对应的*A*的幂的次数

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

ans=ans\*t%C;

}

**return** -1;*//*无解返回*-1*

}

**void** pre() {**for**(**int** i=0;i<=mod;i++)hash[i].f=0,hash[i].num=hash[i].v=-1;}

**int** main()

{

**while**(~scanf(**"%d%d%d"**,&A,&C,&B))

{

**if**(!(A+B+C)) **break**;

**if**(C==1) {puts(**"0"**); **continue**;}

pre(); A%=C; B%=C; *//*数据初始化

ll ans=Shank();

**if**(ans==-1) puts(**"No Solution"**);

**else** printf(**"%lld\n"**,ans);

}

**return** 0;

}

斯特林公式：

n!的位数： **return** ceil((n\*log(**double**(n))-N+0.5\*log(2.0\*n\*PI))/ln\_10);

指数和原根**:**

指数：设m>1，gcd(a,m)=1，使a^x mod m=1 成立的最小的 x，称为 a 对模 m 的指数（阶）。

原根：设m是正整数，a是整数，若 a 模 m 的阶等于 phi(m)，则称 a 为模 m 的一个原根。

定理：若 p 为素数，则 p 必有原根，且模 p 的原根个数为 phi(p-1)。

定理：若模 m 有原根，那么它共有 phi ( phi (m) ) 个原根。

求模素数 p 原根的方法：对 p-1 进行素因子分解，记pi为p-1的第i个因子，若恒有a^((p-1)/pi) mod p ≠ 1 成立，则 a 就是 p 的原根。（对于合数求原根，只需把 p-1 换成 phi(p) 即可）

Code:

**int** p,cnt,len,pr[50010],pri[50010],ispri[50010];

**int** gcd(**int** a,**int** b) {**return** !b?a:gcd(b,a%b);}

**int** fast(**int** a,**int** b,**int** mod) {**int** sum=1;**for**(;b;b>>=1,a=a\*a%mod)**if**(b&1)sum\*=a;**return** sum;}

**void** get()

{

**for**(**int** i=2;i<=50000;i++)

{

**if**(!ispri[i]) pri[++cnt]=i;

**for**(**int** j=1;j<=cnt&&pri[j]\*i<=50000;j++)

{

ispri[pri[j]\*i]=1;

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

}

}

**int** tmp=p-1;

**for**(**int** i=1;i<=cnt;i++)

{

**if**(tmp%pri[i]==0) pr[++len]=pri[i];

**while**(tmp%pri[i]==0) tmp/=pri[i];

}

**if**(tmp>1) pr[++len]=tmp;

}

**bool** check(**int** d)

{

**if**(gcd(p,d)!=1) **return** 0;

**for**(**int** i=1;i<=len;i++) **if**(fast(d,(p-1)/pr[i],p)==1) **return** 0;

**return** 1;

}

**int** main()

{

scanf(**"%d"**,&p);get();

**for**(**int** i=2;i<p;i++) **if**(check(i)) {printf(**"%d\n"**,i); **break**;}

**return** 0;

}

反素数表：

#include<cstdio>

#include<algorithm>

using namespace std;

int maxn,L,R,f,ans[20000010],p[]={0,2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53};

void dfs(int id,int now,int tot)

{

ans[now]=tot;

for(int i=1;now\*p[id]<=R;++i) dfs(id+1,now\*=p[id],tot\*(i+1));

}

int main()

{

scanf("%d%d",&L,&R);

dfs(1,1,1);

for(int i=1;i<L;++i) maxn=max(maxn,ans[i]);

for(int i=L;i<=R;++i)

if(ans[i]>maxn){

maxn=ans[i];

f?printf(","):f=1;

printf("%d",i);

}

if(!f) puts("NO");

return 0;

}

**BM**线性递推式：

#include <bits/stdc++.h>

using namespace std;

#define rep(i, a, n) for (ll i=a;i<n;i++)

#define SZ(x) ((ll)(x).size())

typedef long long ll;

const ll mod = 1000000007;

ll qpow(ll a, ll b) {

ll res = 1;

a %= mod;

for (; b; b >>= 1) {

if (b & 1)res = res \* a % mod;

a = a \* a % mod;

}

return res;

}

ll n;

namespace linear\_seq {

const ll N = 100010;

ll res[N], base[N], \_c[N], \_md[N];

vector<ll> Md;

void mul(ll \*a, const ll \*b, ll k) {

rep(i, 0, k + k) \_c[i] = 0;

rep(i, 0, k) if (a[i]) rep(j, 0, k) \_c[i + j] = (\_c[i + j] + a[i] \* b[j]) % mod;

for (ll i = k + k - 1; i >= k; i--)

if (\_c[i])

rep(j, 0, SZ(Md)) \_c[i - k + Md[j]] = (\_c[i - k + Md[j]] - \_c[i] \* \_md[Md[j]]) % mod;

rep(i, 0, k) a[i] = \_c[i];

}

ll solve(ll n, vector<ll> a, vector<ll> b) {

ll ans = 0, pnt = 0;

ll k = SZ(a);

rep(i, 0, k) \_md[k - 1 - i] = -a[i];

\_md[k] = 1;

Md.clear();

rep(i, 0, k) if (\_md[i] != 0) Md.push\_back(i);

rep(i, 0, k) res[i] = base[i] = 0;

res[0] = 1;

while (1ll << pnt <= n) pnt++;

for (ll p = pnt; p >= 0; p--) {

mul(res, res, k);

if (n >> p & 1) {

for (ll i = k - 1; i >= 0; i--) res[i + 1] = res[i];

res[0] = 0;

rep(j, 0, SZ(Md)) res[Md[j]] = (res[Md[j]] - res[k] \* \_md[Md[j]]) % mod;

}

}

rep(i, 0, k) ans = (ans + res[i] \* b[i]) % mod;

if (ans < 0) ans += mod;

return ans;

}

vector<ll> BM(vector<ll> s) {

vector<ll> C(1, 1), B(1, 1);

ll L = 0, m = 1, b = 1;

rep(n, 0, SZ(s)) {

ll d = 0;

rep(i, 0, L + 1) d = (d + (ll) C[i] \* s[n - i]) % mod;

if (d == 0) ++m;

else if (2 \* L <= n) {

vector<ll> T = C;

ll c = mod - d \* qpow(b, mod - 2) % mod;

while (SZ(C) < SZ(B) + m) C.push\_back(0);

rep(i, 0, SZ(B)) C[i + m] = (C[i + m] + c \* B[i]) % mod;

L = n + 1 - L;

B = T;

b = d;

m = 1;

} else {

ll c = mod - d \* qpow(b, mod - 2) % mod;

while (SZ(C) < SZ(B) + m) C.push\_back(0);

rep(i, 0, SZ(B)) C[i + m] = (C[i + m] + c \* B[i]) % mod;

++m;

}

}

return C;

}

vector<ll> temp;

void init(vector<ll> a) {

temp = BM(a);

temp.erase(temp.begin());

rep(i, 0, SZ(temp))temp[i] = (mod - temp[i]) % mod;

}

ll gao(vector<ll> a, ll n) {

return solve(n, temp, vector<ll>(a.begin(), a.begin() + SZ(temp)));

}

};

using namespace linear\_seq;

int main() {

vector<ll> v;

v.push\_back(3);

v.push\_back(9);

v.push\_back(20);

v.push\_back(46);

v.push\_back(106);

v.push\_back(244);

v.push\_back(560);

v.push\_back(1286);

v.push\_back(2956);

v.push\_back(6794);

init(v);

ll T;

scanf("%lld", &T);

while (T--) {

ll n;

scanf("%lld", &n);

printf("%lld\n", gao(v, n - 1));

}

return 0;

}