# 数学

## 进制转换

```
//luogu1017
#include<cstdio>
#include<iostream>
using namespace std;
int m,n;
void change(int a){
	int k;
	k=a%m;
	a=a/m;
	if(k<0){
		k-=m;
		a++;
	}
	if (a!=0) change(a);
	if(k>9) printf("%c",k-10+'A');
	else printf("%d",k);
}
int main()
{
	scanf("%d%d",&n,&m);
	printf("%d=",n);
	change(n);
	printf("(base%d)",m);
	return 0;
}
```

## 埃筛

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int MAXN=1e5;
bool ipr[MAXN+20];
int cnt,pri[MAXN/5];
void prime(){//埃式筛法
	int N=sqrt(MAXN)+0.5,mul;
	memset(ipr,true,sizeof(ipr));
	ipr[1]=false;
	for(int i=2;i<=N;i++){
		if(ipr[i]==true){
			i==2?mul=1:mul=2;
			for(int j=i*i;j<=MAXN;j+=i*mul){
				ipr[j]=false;
			}
		}
	}
```

```
        for(int i=2;i<=MAXN;i++){
                if(ipr[i]==true){
                        pri[++cnt]=i;
                }
        }
}
int main(){
        freopen("stdout.in","w",stdout);
        clock_t start = clock();
        prime();
        clock_t ends = clock();
        cout <<"Running Time : "<<(double)(ends - start)/ CLOCKS_PER_SEC << endl;
        //cout<<cnt<<endl;
        cout<<"int pri[1020]={";
        for(int i=1;i<=1020;i++){
                cout<<pri[i]<<",";
        }
        cout<<"};\n";
        return 0;
}
```

## 欧拉筛

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN=1e5;
bool ipr[MAXN+20];
int cnt,pri[MAXN/5];
void prime(){//欧拉筛
        memset(ipr,true,sizeof(ipr));
    ipr[1]=false;
    for(int i=2;i<=MAXN;i++){
        if(ipr[i])  pri[++cnt]=i;
        for(int j=1;j<=cnt&&pri[j]<=MAXN/i;j++){
            ipr[i*pri[j]]=false;
            if(!i%pri[j])  break;
        }
    }
}
int main(){
        freopen("stdout.in","w",stdout);
    clock_t start = clock();
        prime();
        clock_t ends = clock();
        cout<<"int pri[1020]={";
        for(int i=1;i<=1020;i++){
                cout<<pri[i]<<",";
        }
        cout<<"};\n";
        cout <<"Running Time : "<<(double)(ends - start)/ CLOCKS_PER_SEC << endl;
        return 0;
}
```

## 计算系数

```cpp
//luogu1313
#include<iostream>
#include<cstdio>
using namespace std;
int f[1020][1020];
int a,b,k,n,m;
int speedm(int x,int c,int p){
    int cur=1,k=x;
    while(c){
        if(c&1){
            cur=cur*k%p;
        }
        k=k*k%p;
        c>>=1;
    }
    return cur;
}
int dfs(int h,int l){
    if(h<l) return 0;
    if(h==0) return 0;
    if(h==l||l==0) return f[h][l]=1;
    if(l==1||l==h-1) return f[h][l]=h;
    int a=(f[h-1][l]==0?f[h-1][l]=dfs(h-1,l):f[h-1][l]);
    int b=(f[h-1][l-1]==0?f[h-1][l-1]=dfs(h-1,l-1):f[h-1][l-1]);
    return f[h][l]=a+b□007;
}
int main(){
    int luck=10007;
    scanf("%d%d%d%d%d",&a,&b,&k,&n,&m);
    int a1=speedm(b%luck,m,10007),a2=speedm(a%luck,n,10007);
    int a3=dfs(k,n)%luck;
    //cout<<a1<<a2<<a3;
    cout<<(((a3*a2)%luck)*a1)%luck;
    return 0;
}
```

## 可靠快速幂

```cpp
#include<bits/stdc++.h>
#define ll long long
#define ld long double
using namespace std;
ll mod=((1LL<<62)-1)|(1LL<<62);//位运算求long long上限
ll mul_mod(ll a,ll b,ll p){//快速乘，这里用不到
        ll ret=a*b-(ll)(a*(ld)b/p+0.5)*p;
        return ret>=0?ret:(ret+p)%p;
}
ll lowspeed(ll a,ll b,ll p){
        ll cur=a,ans=0;
        while(b){
                if(b&1) ans=(ans+cur)%p;
                cur=(cur+cur)%p;
                b>>=1;
        }
        return ans%p;
}
ll speed(ll a,ll b,ll p){
        ll cur=a,ans=1;
        while(b){
                if(b&1) ans=lowspeed(ans,cur,p)%p;
                cur=lowspeed(cur,cur,p)%p;
                b>>=1;
        }
        return ans%p;
}
int main(){
        cout<<mod<<endl;
        cout<<mul_mod(1000000000,1555555555555,1000000007)<<endl;
        cout<<speed(2,61,mod)<<endl;
        return 0;
}
```

## 逆元 阶乘 组合数

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define db double
#define mod 10007
const int MAXN=3e6;
ll n,_inv[MAXN+20];
ll gcd(ll a,ll b){
        return b==0?a:gcd(b,a%b);
}
ll speed(ll a,ll b,ll p){//若mod为质数，逆元speed(b,mod-2,mod)
        ll cur=a,ans=1;
        while(b){
                if(b&1) ans=ans*cur%p;
                cur=cur*cur%p;
                b>>=1;
        }
        return ans%p;
}
ll exgcd(ll a,ll b,ll &x,ll &y){//扩展欧几里得算法，使用时注意+mod再%mod
    if(b==0){//递归边界
        x=1;y=0;
        return a;
    }
    ll ret=exgcd(b,a%b,x,y);
    ll tmp=y;//求解原x,y
    y=x-a/b*y;
    x=tmp;
    return ret;//返回gcd
}
void pre(ll p){
	_inv[0]=_inv[1]=1;
	for(int i=2;i<=MAXN;i++){
		_inv[i]=((p-p/i)*_inv[p%i])%p;
	}
}
ll Scomb(ll _n,ll _m,ll p){//SmallCombination n,m可以线性求出
	if(_m==0) return 1;
	ll ans=1,tmp=1;
	for(ll i=_m+1;i<=_n;i++){
		ans=(ans*i)%p;
	}
	for(ll i=1;i<=_n-_m;i++){
		tmp=(tmp*i)%p;
	}
	//cout<<tmp<<endl;
	return ans*inv(tmp%p,p)%p;
}
ll Bcomb(ll _n,ll _m,ll p){//BigCombination
	if(_n<p&&_m<p) return Scomb(_n,_m,p)%p;
	return Bcomb(_n/p,_m/p,p)*Scomb(_n%p,_m%p,p)%p;
}
int main(){
	pre(mod);
	freopen("a.txt","w",stdout);
	int len=1e9;
	printf("{");
	ll cur=1,p=1e9+7;
	for(ll i=1;i<=len;i++){
		cur=(cur*i)%p;
		if(i%(len/100)==0){
			printf("%lld",cur);
			if(i!=len) printf(",");
		}
	}
	printf("};");
	return 0;
}
```

## 矩阵快速幂

```
#include<iostream>
#include<cstdio>
#include<cstring>
#define ll long long
using namespace std;
long long n,mod=1000000007;
long long m;
struct jz{
	long long a[120][120];
};
jz s;
jz fz(jz &x){
	memset(x.a,0,sizeof(x.a));
	for(int i=1;i<=n;i++){
		x.a[i][i]=1;
	}
	return x;
}
jz cf(jz x,jz y,ll p){
	jz neww;
	memset(neww.a,0,sizeof(neww.a));
	for(int i=1;i<=n;i++){
		for(int j=1;j<=n;j++){
			for(int k=1;k<=n;k++){
				neww.a[i][j]=(x.a[i][k]*y.a[k][j])%p+neww.a[i][j
				neww.a[i][j]%=p;
			}
		}
	}
	return neww;
}
jz speed(jz x,ll b,ll p){
	jz cur=x,ans=fz(ans);
	while(b){
		if(b&1){
			ans=cf(ans,cur,p);
		}
		cur=cf(cur,cur,p);
		b>>=1;
	}
	return ans;
}
int main(){
	scanf("%lld%lld",&n,&m);
	for(int i=1;i<=n;i++){
		for(int j=1;j<=n;j++){
			scanf("%lld",&s.a[i][j]);
		}
	}
	jz x=speed(s,m,mod);
	for(int i=1;i<=n;i++){
		for(int j=1;j<=n;j++){
			printf("%lld ",x.a[i][j]);
		}
		printf("\n");
	}
	return 0;
}
```

## 高精度

```
#include<iostream>
#include<sstream>
#include<algorithm>
```

```cpp
#include<cstring>
#include<iomanip>
#include<vector>
#include<cmath>
#include<ctime>
#include<stack>
using namespace std;
struct Wint:vector<int>//用标准库vector做基类，完美解决位数问题，同时更易于实现
{
    //将低精度转高精度的初始化，可以自动被编译器调用
    //因此无需单独写高精度数和低精度数的运算函数，十分方便
    Wint(int n=0)//默认初始化为0，但0的保存形式为空
    {
        push_back(n);
        check();
    }
    Wint& check()//在各类运算中经常用到的进位小函数，不妨内置
    {
        while(!empty()&&!back())pop_back();//去除最高位可能存在的0
        if(empty())return *this;
        for(int i=1; i<size(); ++i)//处理进位
        {
            (*this)[i]+=(*this)[i-1]/10;
            (*this)[i-1]%=10;
        }
        while(back()>=10)
        {
            push_back(back()/10);
            (*this)[size()-2]%=10;
        }
        return *this;//为使用方便，将进位后的自身返回引用
    }
};
//输入输出
istream& operator>>(istream &is,Wint &n)
{
    string s;
    is>>s;
    n.clear();
    for(int i=s.size()-1; i>=0; --i)n.push_back(s[i]-'0');
    return is;
}
ostream& operator<<(ostream &os,const Wint &n)
{
    if(n.empty())os<<0;
    for(int i=n.size()-1; i>=0; --i)os<<n[i];
    return os;
}
//比较，只需要写两个，其他的直接代入即可
//常量引用当参数，避免拷贝更高效
bool operator!=(const Wint &a,const Wint &b)
{
    if(a.size()!=b.size())return 1;
    for(int i=a.size()-1; i>=0; --i)
        if(a[i]!=b[i])return 1;
    return 0;
}
bool operator==(const Wint &a,const Wint &b)
{
    return !(a!=b);
}
bool operator<(const Wint &a,const Wint &b)
{
    if(a.size()!=b.size())return a.size()<b.size();
    for(int i=a.size()-1; i>=0; --i)
        if(a[i]!=b[i])return a[i]<b[i];
    return 0;
}
bool operator>(const Wint &a,const Wint &b)
{
    return b<a;
}
bool operator<=(const Wint &a,const Wint &b)
{
    return !(a>b);
}
bool operator>=(const Wint &a,const Wint &b)
{
    return !(a<b);
}
//加法，先实现+=，这样更简洁高效
Wint& operator+=(Wint &a,const Wint &b)
{
    if(a.size()<b.size())a.resize(b.size());
    for(int i=0; i!=b.size(); ++i)a[i]+=b[i];
    return a.check();
}
Wint operator+(Wint a,const Wint &b)
{
    return a+=b;
}
//减法，返回差的绝对值，由于后面有交换，故参数不用引用
Wint& operator-=(Wint &a,Wint b)
{
    if(a<b)swap(a,b);
    for(int i=0; i!=b.size(); a[i]-=b[i],++i)
        if(a[i]<b[i])//需要借位
        {
            int j=i+1;
            while(!a[j])++j;
            while(j>i)
            {
                --a[j];
                a[--j]+=10;
            }
        }
    return a.check();
}
Wint operator-(Wint a,const Wint &b)
{
```

```cpp
    return a-=b;
}
//乘法不能先实现*=，原因自己想
Wint operator*(const Wint &a,const Wint &b)
{
    Wint n;
    n.assign(a.size()+b.size()-1,0);
    for(int i=0; i!=a.size(); ++i)
        for(int j=0; j!=b.size(); ++j)
            n[i+j]+=a[i]*b[j];
    return n.check();
}
Wint& operator*=(Wint &a,const Wint &b)
{
    return a=a*b;
}
//除法和取模先实现一个带余除法函数
Wint divmod(Wint &a,const Wint &b)
{
    Wint ans;
    for(int t=a.size()-b.size(); a>=b; --t)
    {
        Wint d;
        d.assign(t+1,0);
        d.back()=1;
        Wint c=b*d;
        while(a>=c)
        {
            a-=c;
            ans+=d;
        }
    }
    return ans;
}
Wint operator/(Wint a,const Wint &b)
{
    return divmod(a,b);
}
Wint& operator/=(Wint &a,const Wint &b)
{
    return a=a/b;
}
Wint& operator%=(Wint &a,const Wint &b)
{
    divmod(a,b);
    return a;
}
Wint operator%(Wint a,const Wint &b)
{
    return a%=b;
}
Wint pow(Wint n,Wint k)
{
        Wint cur=n,ans=1;
        while(k!=0){
                if(k%2==1) ans*=cur;
                cur*=cur;
                k=k/2;
        }
        return ans;
}
int main()
{
        Wint p;
        cin>>p;
        Wint ans=pow(Wint(2),p)-1;
        cout<<ans.size()<<endl;
        cout<<ans<<endl;
        return 0;
}
```