

# 数学

## 进制转换

```
1 //luogu1017
2 #include<cstdio>
3 #include<iostream>
4 using namespace std;
5 int m,n;
6 void change(int a){
7     int k;
8     k=a%m;
9     a=a/m;
10    if(k<0){
11        k-=m;
12        a++;
13    }
14    if (a!=0) change(a);
15    if(k>9) printf("%c",k-10+'A');
16    else printf("%d",k);
17 }
18 int main()
19 {
20     scanf("%d%d",&n,&m);
21     printf("%d=",n);
22     change(n);
23     printf("(base%d)",m);
24     return 0;
25 }
```

## 埃筛

```
1 #include<bits/stdc++.h>
2 #define ll long long
3 using namespace std;
4 const int MAXN=1e5;
5 bool ipr[MAXN+20];
6 int cnt,pri[MAXN/5];
7 void prime(){//埃式筛法
8     int N=sqrt(MAXN)+0.5,mul;
9     memset(ipr,true,sizeof(ipr));
10    ipr[1]=false;
11    for(int i=2;i<=N;i++){
12        if(ipr[i]==true){
13            i==2?mul=1:mul=2;
14            for(int j=i*i;j<=MAXN;j+=i*mul){
15                ipr[j]=false;
16            }
17        }
18    }
```

```

19         for(int i=2;i<=MAXN;i++){
20             if(ipr[i]==true){
21                 pri[++cnt]=i;
22             }
23         }
24     }
25     int main(){
26         freopen("stdout.in","w",stdout);
27         clock_t start = clock();
28         prime();
29         clock_t ends = clock();
30         cout <<"Running Time : "<<(double)(ends - start)/ CLOCKS_PER_SEC << endl;
31         //cout<<cnt<<endl;
32         cout<<"int pri[1020]={";
33         for(int i=1;i<=1020;i++){
34             cout<<pri[i]<<",";
35         }
36         cout<<"};\n";
37         return 0;
38     }

```

## 欧拉筛

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int MAXN=1e5;
4  bool ipr[MAXN+20];
5  int cnt,pri[MAXN/5];
6  void prime(){//欧拉筛
7      memset(ipr,true,sizeof(ipr));
8      ipr[1]=false;
9      for(int i=2;i<=MAXN;i++){
10         if(ipr[i]) pri[++cnt]=i;
11         for(int j=1;j<=cnt&&pri[j]<=MAXN/i;j++){
12             ipr[i*pri[j]]=false;
13             if(!i%pri[j]) break;
14         }
15     }
16 }
17 int main(){
18     freopen("stdout.in","w",stdout);
19     clock_t start = clock();
20     prime();
21     clock_t ends = clock();
22     cout<<"int pri[1020]={";
23     for(int i=1;i<=1020;i++){
24         cout<<pri[i]<<",";
25     }
26     cout<<"};\n";
27     cout <<"Running Time : "<<(double)(ends - start)/ CLOCKS_PER_SEC << endl;
28     return 0;
29 }

```

## 计算系数

```
1 //luogu1313
2 #include<iostream>
3 #include<cstdio>
4 using namespace std;
5 int f[1020][1020];
6 int a,b,k,n,m;
7 int speedm(int x,int c,int p){
8     int cur=1,k=x;
9     while(c){
10         if(c&1){
11             cur=cur*k%p;
12         }
13         k=k*k%p;
14         c>>=1;
15     }
16     return cur;
17 }
18 int dfs(int h,int l){
19     if(h<l) return 0;
20     if(h==0) return 0;
21     if(h==1||l==0) return f[h][l]=1;
22     if(l==1||l==h-1) return f[h][l]=h;
23     int a=(f[h-1][l]==0?f[h-1][l]=dfs(h-1,l):f[h-1][l]);
24     int b=(f[h-1][l-1]==0?f[h-1][l-1]=dfs(h-1,l-1):f[h-1][l-1]);
25     return f[h][l]=a+b%007;
26 }
27 int main(){
28     int luck=10007;
29     scanf("%d%d%d%d%d",&a,&b,&k,&n,&m);
30     int a1=speedm(b%luck,m,10007),a2=speedm(a%luck,n,10007);
31     int a3=dfs(k,n)%luck;
32     //cout<<a1<<a2<<a3;
33     cout<<(((a3*a2)%luck)*a1)%luck;
34     return 0;
35 }
```

## 可靠快速幂

```
1 #include<bits/stdc++.h>
2 #define ll long long
3 #define ld long double
4 using namespace std;
5 ll mod=((1LL<<62)-1)|(1LL<<62);//位运算求long long上限
6 ll mul_mod(ll a,ll b,ll p){//快速乘，这里用不到
7     ll ret=a*b-(ll)(a*(ld)b/p+0.5)*p;
8     return ret>=0?ret:(ret+p)%p;
9 }
10 ll lowspeed(ll a,ll b,ll p){
11     ll cur=a,ans=0;
12     while(b){
```

```

13         if(b&1) ans=(ans+cur)%p;
14         cur=(cur+cur)%p;
15         b>>=1;
16     }
17     return ans%p;
18 }
19 ll speed(ll a,ll b,ll p){
20     ll cur=a,ans=1;
21     while(b){
22         if(b&1) ans=lowspeed(ans,cur,p)%p;
23         cur=lowspeed(cur,cur,p)%p;
24         b>>=1;
25     }
26     return ans%p;
27 }
28 int main(){
29     cout<<mod<<endl;
30     cout<<mul_mod(1000000000,155555555555,1000000007)<<endl;
31     cout<<speed(2,61,mod)<<endl;
32     return 0;
33 }

```

## 逆元 阶乘 组合数

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define ll long long
4  #define db double
5  #define mod 10007
6  const int MAXN=3e6;
7  ll n,_inv[MAXN+20];
8  ll gcd(ll a,ll b){
9      return b==0?a:gcd(b,a%b);
10 }
11 ll speed(ll a,ll b,ll p){//若mod为质数，逆元speed(b,mod-2,mod)
12     ll cur=a,ans=1;
13     while(b){
14         if(b&1) ans=ans*cur%p;
15         cur=cur*cur%p;
16         b>>=1;
17     }
18     return ans%p;
19 }
20 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
21     if(b==0){//递归边界
22         x=1;y=0;
23         return a;
24     }
25     ll ret=exgcd(b,a%b,x,y);
26     ll tmp=y;//求解原x,y
27     y=x-a/b*y;
28     x=tmp;
29     return ret;//返回gcd

```

```

30 }
31 void pre(ll p){
32     _inv[0]=_inv[1]=1;
33     for(int i=2;i<=MAXN;i++){
34         _inv[i]=((p-p/i)*_inv[p%i])%p;
35     }
36 }
37 ll Scomb(ll _n,ll _m,ll p){//SmallCombination n,m可以线性求出
38     if(_m==0) return 1;
39     ll ans=1,tmp=1;
40     for(ll i=_m+1;i<=_n;i++){
41         ans=(ans*i)%p;
42     }
43     for(ll i=1;i<=_n-_m;i++){
44         tmp=(tmp*i)%p;
45     }
46     //cout<<tmp<<endl;
47     return ans*inv(tmp%p,p)%p;
48 }
49 ll Bcomb(ll _n,ll _m,ll p){//BigCombination
50     if(_n<p&&_m<p) return Scomb(_n,_m,p)%p;
51     return Bcomb(_n/p,_m/p,p)*Scomb(_n%p,_m%p,p)%p;
52 }
53 int main(){
54     pre(mod);
55     freopen("a.txt","w",stdout);
56     int len=1e9;
57     printf("{");
58     ll cur=1,p=1e9+7;
59     for(ll i=1;i<=len;i++){
60         cur=(cur*i)%p;
61         if(i%(len/100)==0){
62             printf("%lld",cur);
63             if(i!=len) printf(",");
64         }
65     }
66     printf("}");
67     return 0;
68 }

```

## 矩阵快速幂

```

1  #include<iostream>
2  #include<cstdio>
3  #include<cstring>
4  #define ll long long
5  using namespace std;
6  long long n,mod=1000000007;
7  long long m;
8  struct jz{
9      long long a[120][120];
10 };
11 jz s;

```

```

12 jz fz(jz &x){
13     memset(x.a,0,sizeof(x.a));
14     for(int i=1;i<=n;i++){
15         x.a[i][i]=1;
16     }
17     return x;
18 }
19 jz cf(jz x,jz y,ll p){
20     jz neww;
21     memset(neww.a,0,sizeof(neww.a));
22     for(int i=1;i<=n;i++){
23         for(int j=1;j<=n;j++){
24             for(int k=1;k<=n;k++){
25                 neww.a[i][j]=(x.a[i][k]*y.a[k][j])%p+neww.a[i][j];
26                 neww.a[i][j]%=p;
27             }
28         }
29     }
30     return neww;
31 }
32 jz speed(jz x,ll b,ll p){
33     jz cur=x,ans=fz(ans);
34     while(b){
35         if(b&1){
36             ans=cf(ans,cur,p);
37         }
38         cur=cf(cur,cur,p);
39         b>>=1;
40     }
41     return ans;
42 }
43 int main(){
44     scanf("%lld%lld",&n,&m);
45     for(int i=1;i<=n;i++){
46         for(int j=1;j<=n;j++){
47             scanf("%lld",&s.a[i][j]);
48         }
49     }
50     jz x=speed(s,m,mod);
51     for(int i=1;i<=n;i++){
52         for(int j=1;j<=n;j++){
53             printf("%lld ",x.a[i][j]);
54         }
55         printf("\n");
56     }
57     return 0;
58 }

```

## 高精度

```

1 #include<iostream>
2 #include<sstream>
3 #include<algorithm>

```

```

4  #include<cstring>
5  #include<iomanip>
6  #include<vector>
7  #include<cmath>
8  #include<ctime>
9  #include<stack>
10 using namespace std;
11 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
12 {
13     //将低精度转高精度的初始化，可以自动被编译器调用
14     //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
15     Wint(int n=0){//默认初始化为0，但0的保存形式为空
16     {
17         push_back(n);
18         check();
19     }
20     Wint& check();//在各类运算中经常用到的进位小函数，不妨内置
21     {
22         while(!empty()&&!back())pop_back();//去除最高位可能存在的0
23         if(empty())return *this;
24         for(int i=1; i<size(); ++i)//处理进位
25         {
26             (*this)[i]+=(*this)[i-1]/10;
27             (*this)[i-1]%=10;
28         }
29         while(back()>=10)
30         {
31             push_back(back()/10);
32             (*this)[size()-2]%=10;
33         }
34         return *this;//为使用方便，将进位后的自身返回引用
35     }
36 };
37 //输入输出
38 istream& operator>>(istream &is,Wint &n)
39 {
40     string s;
41     is>>s;
42     n.clear();
43     for(int i=s.size()-1; i>=0; --i)n.push_back(s[i]-'0');
44     return is;
45 }
46 ostream& operator<<(ostream &os,const Wint &n)
47 {
48     if(n.empty())os<<0;
49     for(int i=n.size()-1; i>=0; --i)os<<n[i];
50     return os;
51 }
52 //比较，只需要写两个，其他的直接代入即可
53 //常量引用当参数，避免拷贝更高效
54 bool operator!=(const Wint &a,const Wint &b)
55 {
56     if(a.size()!=b.size())return 1;
57     for(int i=a.size()-1; i>=0; --i)
58         if(a[i]!=b[i])return 1;

```

```

59     return 0;
60 }
61 bool operator==(const Wint &a,const Wint &b)
62 {
63     return !(a!=b);
64 }
65 bool operator<(const Wint &a,const Wint &b)
66 {
67     if(a.size()!=b.size())return a.size()<b.size();
68     for(int i=a.size()-1; i>=0; --i)
69         if(a[i]!=b[i])return a[i]<b[i];
70     return 0;
71 }
72 bool operator>(const Wint &a,const Wint &b)
73 {
74     return b<a;
75 }
76 bool operator<=(const Wint &a,const Wint &b)
77 {
78     return !(a>b);
79 }
80 bool operator>=(const Wint &a,const Wint &b)
81 {
82     return !(a<b);
83 }
84 //加法, 先实现+=, 这样更简洁高效
85 Wint& operator+=(Wint &a,const Wint &b)
86 {
87     if(a.size()<b.size())a.resize(b.size());
88     for(int i=0; i!=b.size(); ++i)a[i]+=b[i];
89     return a.check();
90 }
91 Wint operator+(Wint a,const Wint &b)
92 {
93     return a+=b;
94 }
95 //减法, 返回差的绝对值, 由于后面有交换, 故参数不用引用
96 Wint& operator-=(Wint &a,Wint b)
97 {
98     if(a<b)swap(a,b);
99     for(int i=0; i!=b.size(); a[i]-=b[i],++i)
100         if(a[i]<b[i])//需要借位
101         {
102             int j=i+1;
103             while(!a[j])++j;
104             while(j>i)
105             {
106                 --a[j];
107                 a[--j]+=10;
108             }
109         }
110     return a.check();
111 }
112 Wint operator-(Wint a,const Wint &b)
113 {

```



```

114     return a-=b;
115 }
116 //乘法不能先实现*=, 原因自己想
117 Wint operator*(const Wint &a,const Wint &b)
118 {
119     Wint n;
120     n.assign(a.size()+b.size()-1,0);
121     for(int i=0; i!=a.size(); ++i)
122         for(int j=0; j!=b.size(); ++j)
123             n[i+j]+=a[i]*b[j];
124     return n.check();
125 }
126 Wint& operator*=(Wint &a,const Wint &b)
127 {
128     return a=a*b;
129 }
130 //除法和取模先实现一个带余除法函数
131 Wint divmod(Wint &a,const Wint &b)
132 {
133     Wint ans;
134     for(int t=a.size()-b.size(); a>=b; --t)
135     {
136         Wint d;
137         d.assign(t+1,0);
138         d.back()=1;
139         Wint c=b*d;
140         while(a>=c)
141         {
142             a-=c;
143             ans+=d;
144         }
145     }
146     return ans;
147 }
148 Wint operator/(Wint a,const Wint &b)
149 {
150     return divmod(a,b);
151 }
152 Wint& operator/=(Wint &a,const Wint &b)
153 {
154     return a=a/b;
155 }
156 Wint& operator%=(Wint &a,const Wint &b)
157 {
158     divmod(a,b);
159     return a;
160 }
161 Wint operator%(Wint a,const Wint &b)
162 {
163     return a%=b;
164 }
165 Wint pow(Wint n,Wint k)
166 {
167     Wint cur=n,ans=1;
168     while(k!=0){

```

```
169         if(k%2==1) ans*=cur;
170         cur*=cur;
171         k=k/2;
172     }
173     return ans;
174 }
175 int main()
176 {
177     Wint p;
178     cin>>p;
179     Wint ans=pow(Wint(2),p)-1;
180     cout<<ans.size()<<endl;
181     cout<<ans<<endl;
182     return 0;
183 }
```