## GCD&LCM

ll gcd(ll a,ll b) {

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

}

ll lcm(ll a,ll b) {

return a/gcd(a,b)\*b;

}

## 等比数列求和

ll Geo(ll head, ll comm, ll ns, ll mod) {

if(ns==1) return head;

if(comm==1) return ((head%mod)\*(ns%mod))%mod;

return head\*((pow\_mod(comm,ns,mod)-1)\*(inv(comm-1,mod)%mod))%mod;

}

## 素数筛

int prime[maxn];

bool is\_prime[maxn];

int sieve(int n){

int p = 0;

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

is\_prime[i] = true;

is\_prime[0] = is\_prime[1] = false;

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

if(is\_prime[i]){

prime[p++] = i;

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

is\_prime[j] = false;

}

}

return p; // 返回素数个数

}

## 约数个数定理

ll factor\_num(ll n){

if(is\_prime[n]) return 2;

ll ans = 1, t = n, ic, tmp;

for(ll pos = 0;prime[pos]<=t;pos++){

ic = 0,tmp = prime[pos];

while(n%tmp==0){

n/=tmp;

ic++;

}

ans\*=ic+1;

}

return ans;

}

## 约数和

ll factor\_sum(ll n){

ll ans = 1, t = n, ic, tmp;

for(ll pos = 0;prime[pos]<=t;pos++){

ic = 0,tmp = prime[pos];

while(n%tmp==0){

n/=tmp;

ic++;

}

if(ic>0) ans = ans \* Geo(1,tmp,ic+1,mod) % mod;

}

return ans;

}

## 组合数

Cmn n>=m m在上,n在下

ll C(ll n,ll m){

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

ll res = 1;

for(ll i=1,j=n;i<=k;i++,j--){

res = ((res\*j)%mod)\*inv(i,mod)%mod;

}

return res;

}

## 扩展欧几里德

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

{

if(!b) { d = a; x = 1; y = 0; }

else{ exgcd(b, a%b, d, y, x); y -= x\*(a/b); }

}

## 乘法逆元

ll inv(ll a, ll p)

{

ll d, x, y;

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

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

}

## 快速幂

ll pow\_mod(ll a,ll n,ll m){

if(n==0) return 1;

ll x = pow\_mod(a,n/2,m);

ll ans = x\*x%m;

if(n%2==1) ans = ans\*a%m;

return ans;

}

## 快速幂取模

int power2(int a, int b, int c)

{

int res = 1;

a %= c;

while (b)

{

if (b & 1)

res = (res \* a) % c;

a = (a \* a) % c;

b >>= 1;

}

return res;

}

## 等比数列求和

ll Geo(ll head, ll comm, ll ns, ll mod){

if(ns==1) return head;

if(comm==1) return ((head%mod)\*(ns%mod))%mod;

return head\*((pow\_mod(comm,ns,mod)-1)\*(inv(comm-1,mod)%mod))%mod;

}

## 大数模板

//初始化

void initial(string &a, string &b){

while (a.size()<b.size())a = '0' + a;

while (b.size()<a.size())b = '0' + b;

}

//打印

void print(string &a, string &b){

cout << a << endl;

cout << b << endl;

}

//找出最大的字符串

void findMax(string &a, string &b){

string tmp;

if (a<b){

tmp = b;

b = a;

a = tmp;

}

}

//删除第一个字符'0'

bool del(string &a){

if (a[0] == '0'){

a.erase(0, 1);

return true;

}

else

return false;

}

//删除前面所有的 0

void delAllZroe(string &a){

while (del(a)){

del(a);

};

}

//大数加法

string bigItergeAdd(string a, string b){

initial(a, b);

a = '0' + a;

b = '0' + b;

for (int i = a.size() - 1; i >= 0; i--){

int num1 = a[i] - '0';

int num2 = b[i] - '0';

if (num1 + num2>9){

a[i - 1] = a[i - 1] - '0' + 1 + '0';

a[i] = (num1 + num2) - 10 + '0';

}

else{

a[i] = (num1 + num2) + '0';

}

}

del(a);

// cout<<a<<endl;

return a;

}

//大数减法

string bigItergeSub(string a, string b){

initial(a, b);

findMax(a, b);

for (int i = a.size() - 1; i >= 0; i--){

int num1 = a[i] - '0';

int num2 = b[i] - '0';

if (num1<num2){

a[i - 1] = a[i - 1] - '0' - 1 + '0';

a[i] = (num1 + 10 - num2) + '0';

}

else{

a[i] = (num1 - num2) + '0';

}

}

del(a);

// cout<<a<<endl;

return a;

}

//大数乘法(大数加法实现)

void bigItergeMul(string a, string b){

delAllZroe(a);

delAllZroe(b);

if (a == "" || b == ""){

printf("0\n"); return;

}

initial(a, b);

findMax(a, b);

string res = "0";

int count = 0;

delAllZroe(b);

for (int i = b.size() - 1; i >= 0; i--){

int num1 = b[i] - '0';

if (i != b.size() - 1) a = a + '0';

for (int i = 1; i <= num1; i++){

res = bigItergeAdd(res, a);

}

}

delAllZroe(res);

cout << res << endl;

}

//大数除法

void bigItergeDiv(string a, string b){

initial(a, b);

if (a<b){ cout << "0" << endl; return; }

delAllZroe(b);

string res = "0";

string restmp = "1";

string tmp = b;

for (int i = 1; i<(a.size() - b.size()); i++){

tmp += '0';

restmp += '0';

}

initial(a, b);

while (a >= b){

initial(a, tmp);

if (a >= tmp){

a = bigItergeSub(a, tmp);

res = bigItergeAdd(res, restmp);

}

else{

tmp.erase(tmp.size() - 1);

restmp.erase(restmp.size() - 1);

initial(a, tmp);

if (a >= tmp){

a = bigItergeSub(a, tmp);

res=bigItergeAdd(res, restmp);

}

}

initial(a, b);

}

cout << res << endl;

}