## first section for math from page(2),(3),(4):

1-prime🡪 page(2);

2-seive 🡪page(2);

3-divisor🡪 page(3);

4-prime factorization 🡪page(3);

5-factral 🡪page(5);

6-fibonacci 🡪page();

7)GCD 🡪page();

8)LCM🡪page();

9) permutation🡪page();

***10)combination🡪page();***

***11)divide and conquer🡪page();***

***12)extended Euclidean opservation on gcd 🡪page();***

***13) Pascal Triangle🡪page();***

14)inclusion&excusion

15) invers Moduar 🡪page();

16)

17)

18)

19)

***Section for C\_math***

1)prime

bool ispriem(ll n) {

int n; cin >> n;

if (n == 2)return true;

if (n < 2)return false;

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

if (n % i == 0)return false;

}

return true;

}

bool ispriem(ll n) {

int n; cin >> n;

if (n == 2)return true;

if (n <2||n%2==0)return false;

for (int i = 3; i < n; i+=2) {if (n % i == 0)return false;}

return true;

}

bool ispriem(ll n) {

int n; cin >> n;

if (n == 2)return true;

if (n <2||n%2==0)return false;

ll sq= sqrt(n);

for (int i = 3; i <=sq(n); i+=2) {if (n % i == 0)return false;}

return true;

}

bool ispriem(ll n) {

int n; cin >> n;

if (n == 2)return true;

if (n <2||n%2==0)return false;

for (int i = 3; i\*i <=n; i+=2) {if (n % i == 0)return false;}

return true;

}

***2) sieve***

//back thinking to count prime number in range

int countPrimeNumberInRinge\_sieve(ll n) { //order nlog(n)

vector<bool>isPrime(n + 1, true); //set all of them as prime

isPrime[0] = isPrime[1] = 0; //there si not brime after sqrt

//0,1 not prime

for (int i =2 ; i\*i <= n; i++) {if (isPrime[i]) {for (int j = i \* 2; j <=n; j+=i) {isPrime[j] = 0;}}}

int cnt = 0;

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

if (isPrime[i])cnt++;

}

return cnt;}

***Sieve less than O(nlon(n))***

void sieve\_v2(){

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

prime[i] = 1;

prime[0] = prime[1] = 0;

for (int i = 4; i < N; i += 2){prime[i] = 0;}

for (int i = 3; i \* i <= N; i += 2){

if (prime[i]){

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

prime[j] = 0;

}

}}}

***3)Divisor number***

vector<ll> genrateDivisors(ll n) { //order o(sqrt(n) )

vector<ll>v; // to get the divisor

ll i ;

for (i = 1; i\*i < n; i++) {if (n % i == 0) {v.push\_back(i);v.push\_back( n / i);}}

if (i \* i==n)v.push\_back(i);

return v;

}

-------------------------

void get\_divisor(long long num) {

for (long long i = 1; i \* i <= num; i++) {

if (i \* i == num) {

divisor.push\_back(i);

break;

}

if (num % i == 0) {

divisor.push\_back(i);

divisor.push\_back(num / i);

}

}

}

//بيمسك الرقم من 1 الى( اكس )ويجيب مجموع الارقام الى تقبل عليهم بسرعة

ll get\_sum\_div(ll x){

ll ans = 0;

for (ll left = 1, right; left <= x; left = right + 1){

right = x / (x / left);

ans += (x / left)\*(left + right)\*(right - left + 1) / 2;

}

return ans;

}

***4) prime\_-factorization***

//what about n=0 or n=1;

vector<int>prime\_factorization(ll n) { //max n=1e12 / //(sqrt(n))

vector<int>primes;

for (int i = 2; i \* i <= n; i++) {while (n % i == 0) {primes.push\_back(i); n /= i;}}

if (n > 1)primes.push\_back(n);

return primes;

}

\*

// function to calculate all the prime factors and in form p^n

// count of every prime factor

void factorize(long long n){

int count = 0;

while (!(n % 2)) {

n >>= 1;

count++;

}

if (count)

cout << 2 << " " << count << endl;

for (long long i = 3; i <= sqrt(n); i += 2) {

count = 0;

while (n % i == 0) {

count++; n = n / i;

}

if (count)

cout << i << " " << count << endl;

}

if (n > 2)

cout << n << " " << 1 << endl;

}

// Calculate Prime factors faster in form p^n

//هتنادم فنكشن sieve\_2 ,عرف فوق ارى تخزن فيها شغلى سيف

void Prime\_Factors\_Faster(int n){

vector<int>primes;

sieve\_v2();

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

if (prime[i]){

primes.push\_back(i);}}

// if number is prime

if (prime[n]){

cout << n << " " << 1 << endl;return;}

int idx = 0;

while (idx < primes.size() && primes[idx] \* primes[idx] <= n){

int cnt = 0;

while (n % primes[idx] == 0){

n /= primes[idx];cnt++;}

if (cnt) cout << primes[idx] << " " << cnt << endl;

idx++;}

if (n>1){cout << n << " " << 1 << endl;}}

//16 =2^4 has 5 divisors :2^0,2^1,2^2,2^3,2^4

//p^n has (n+1) divisor ,p: for any prime number

//what about (p1^a)\*(p2^b)? (a+1)\*(b+1)

//E.X 12=2^2\*3^1 has (2+1)\*(1+1)=6 divisor

//12=2^0\*3^0 //has this divisor

//12=2^0\*3^1

//12=2^1\*3^0

//12=2^1\*3^1

//12=2^2\*3^0

//12=2^2\*3^1

//this is 6 divisor

# what about factorization n^power?

// simpy n=p1^a\*p2^b\*p3^c

//then n^z=(p1^az)\*(p2^bz)\*(p3^cz)

//divisor for n=(a+1)\*(b+1)\*(c+1);

//then divior for n^z=(az+1)\*(bz+1)\*(cz+1)

//اكواد عبد الحكيم

// Generates all the prime divisors of the numbers from 1 to n.

vector<int> primeDivs[N];

void generatePrimeDivisors(int n) {

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

if (primeDivs[i].size()) continue;

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

primeDivs[j].push\_back(i);

}}}

// Generates all the divisors of the numbers from 1 to n.

vector<int> divs[N];

void generateDivisors(int n) {

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

for (int j = i; j <= n; j += i)divs[j].push\_back(i);}

}

1. // function to calculate all the prime factors and
2. // count of every prime factor

void factorize(long long n){

int count = 0;

while (!(n % 2)) {n >>= 1;count++;}

if (count)

cout << 2 << " " << count << endl;

for (long long i = 3; i <= sqrt(n); i += 2) {

count = 0;

while (n % i == 0) {

count++; n = n / i;}

if (count)

cout << i << " " << count << endl;}

if (n > 2)cout << n << " " << 1 << endl;

}

***5)factrial***

//information

//fact(0)=fact(1)=1

//exponential mean may by overfloat

//x!%n x>=n 🡪0

//how many digit in factorial like 1000?,9999?,4444?:

// num of digit=1+(int)log(n);

//pow(10,log(n))=x where x is the nalue of number

// where log(a\*b)=log(a)+log(b)

//then log(a!)=log(1)+………..+log(a);

//what about number of bit for factorial ?think number of bit for EX 8 have 4 bit;

//number of bit=1+(int)log2(n)

//very important

// Given prime p and n! ,what is the max x that n! divide by p^x.(mean how many time p in n!.)

فى طريقة حل تانى انا مش هكتبها عشان مش كويسة + هو مكتبهاش

//E.X Exploring [16!,this p=2]

//1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Make it

1

1 2

1 2 2

1 2 2 2

//1 2 3 2 5 6 7 2 9 10 11 12 13 14 15 2

//code

int primeN\_power(int n,int p) { //O(log(n)base b

int pow= 0;

for (int i = p; i < n; i=i\*p) pow += n / i;

return pow;}

// فى امثلة مش مفهومة فى الحتة دى

***6)fib***

int fib(int n) {

if (n <= 1)return n;

return fib(n - 1) + fib(n - 2);}

int interntivfib(int n) {

int a = 0,b = 1,b;

for (int i = 2; i <=n; i++) {int c = a + b;a = b;b = c;}}

***7)GCD***

***// imprortant facts***

If(a%n=0) &(b%n=0) then 🡪🡪🡪🡪(a+b)%n==0 &(a-b)%n==0 as % can be distribute mean

(a-b)%n=(a%n-b%n)%n=(0+0)%n=0

E.X

Gcd(45,10)=gcd(35,10)=gcd(25,10)=gcd(15,10)=gcd(5,10)=gcd(5,5)=gcd(0,5)=5;

Then above =gcd(45,10)=gcd(45%10,10)=gcd(5,10)

Code:

int gcd(int a, int b) {

if (b == 0)return a;

return gcd(b, a% b);

// int gcd(ll a, ll b) { return( (b == 0? a : gcd(b,a % b)) );}

}

***8)LCM***

//prove code last common divisor

//the first multiple of 2 number

//a=2^3 \*7^6

//b=2^5 \*7^2

//to get gcd we need min pow of 2& min pow for 7

//gcd =2^min(3,5)\*7^min(2,6)

//give a,b,&gcd(a,b) how to calc lcm(a,b)

//to get lcm we need max pow of 2& max pow for 7

//lcm=2^max(3,5)\*7^max(2,6)

//what about a\*b=2^(3+5)\*7^(2+6)

//then lcm =a\*b/gcd(a,b);

***//code***

int lcm(int a, int b) {

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

}

***9)permutation***

***Important information***

***//arrange is important mean order***

***//if no repatation***

E.X for set {1,2,3}🡪have permutation (1,2,3),(1,3,2),(2,1,3),(2,3,1),(3,2,1),(3,1,2)🡪n!

How rule?

* rule of product
* In the first choice we can choice n,second choice n-1,thirt choice n-2;
* Then we have n\*(n-1)\*(n-2)…..1way=p(n)=n!

What if we arrange in r element?

* same way have:n\*(n-1)………...\*(n-r+1)
* P(n,r)=n!/(n-r)!

Or ather way to thinking :

* how many combination of r elements,and do for each element r! permutation.
* That is :c(p,r)\*r!;

What about permutation with repatation?

* pr(n)=n^n;

permutation for repation?

EX: AAA or AAABB;

P(n)=n!(c1!\*c2…..);where n🡪number of all didits, c🡪number of repataion of digit=5!/(3!\*2!);

//code to generate all permutation recsion?

vector<int>permutation;

int prem\_cnt = 0, prem\_num = 4;

bool is\_visited[4];

void get\_prem(int i=0) {

if (i == prem\_num) { //finally ew have 24 perutation

prem\_cnt++; //you cna print primtation here

return;

}

for (int j = 0; j < prem\_num; j++) {

if (is\_visited[j]) continue;

permutation.push\_back(j);

is\_visited[j] = 1;

get\_prem(i + 1);

is\_visited[j] = 0;

permutation.pop\_back();

}

}

//code to generate all permutation loop?

vector<int>v = { 1,2,3 };

int cnt = 0;

do {

cnt++;

} while (next\_permutation(all(v))

Code:

1. const int N = 1e6;
2. const int mod = 1e9 + 7;
3. // NPR as the rule may be overfloat
4. ll NpR(ll n, ll r){
5. ll ans = 1;
6. for (ll i = (n - r) + 1; i <= n; i++){
7. ans \*= i;
8. ans %= mod;
9. }
10. return ans;
11. }

***10)combination:***

The number of way to picking r (unorder 🡪like name)element out of n elements;

What calculate?

P(n,r)=c(n,r)\*r!;

n!/(n-r)!=c(n,r)\*r!;

c(n,r)=n!/((n-r)!\*r!));

code:

vector<int>combination;

int cnt = 0,n=29, m= 4;

void get\_comb(int i=0,int last\_visit=0) {if (i == m) { //finally we have 4845 combination cnt++; //you cna print combination here

return;}

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

combination.push\_back(j); get\_comb(i + 1,j); combination.pop\_back();}}

code:

//as the rue may be overflaot

1. const int N = 1e6;
2. const int mod = 1e9 + 7;
3. // NCR
4. ll nCr(ll n, ll r){
5. ll ans = 1;
6. ll div = 1; // r!
7. for (ll i = r + 1; i <= n; i++){
8. ans = ans \* i;
9. ans /= div;
10. div++;
11. }
12. return ans;
13. }

***11)divide and conquer:***

How to calc 5 ^3?simply loop work 3 time; O(power)

other way : //O(log(p)base 2)

* E.X 10^16 make it =(10^8)^2 //event 16
* E.X 10^17 make it =(10^8)^2\*10 //odd 17

// Fast Power

Code: //recursion

int pow(int b, int p) {

if (p == 0)return 1;

int sq = pow(b, p / 2);

sq \*= sq;

if (p % 2)sq\* b;

return sq;}

-------------------------------------------

code x^k by loop

ll power(ll x, ll k){

ll ret = 1;

while (k){

if (k & 1) ret = (ret\*x) % mod;k >>= 1; x = (x\*x) % mod;}return ret;}

what about (x^k)%m?

int pow(ll b, ll m p,ll m) {

if (p == 0)return 1;

ll sq = pow(b, p / 2,m);

sq = (sq\*sq)%m;

if (p % 2)sq=(sq\* b)%m;

return sq;}

What about calculating (a^1+a^2+a^3+a^4+a^5+a^n)?

//lets try evnt power

(a^1+a^2+a^3+a^4+a^5+a^6)= (a^1+a^2+a^3)+( a^4+a^5+a^6)= (a^1=a^2+a^3)+( (a^1\*a^3)+(a^2\*a^3)+(a^3\*a^3))

=(a^1+a^2+a^3)+a^3(a^1+a^2+a^3)= (a^1+a^2+a^3)\*[1+a^3]

=(a^1+a^2+a^3)\*[1+a^3+a^1+a^2-(a^1+a^2)]

//lets try odd power

(a^1+a^2+a^3+a^4+a^5+a^6+a^7)=a^1+ (a^2+a^3+a^4+a^5+a^6+a^7)=a+a(a^1+a^2+a^3+a^4+a^5+a^6)

=a(1+(a^1+a^2+a^3+a^4+a^5+a^6));

Code: //recursion

ll sumPow(ll a,int k) { //returna^1+a^2+a^3...a^k in O(k)

if (k == 0)return 0;

if (k % 2) //saad say==1

return a \* (1 + sumPow(a,k-1));

ll half = sumPow(a, k /2);

return(half \* (1 + half - sumPow(a, k / 2 - 1)));

}

***12) extended Euclidean:***

***// to calculate x and y in equation aX+b\*y=gcd(a,b)***

/ Returns the Bezout's coefficients of the smallest positive linear combination of a and b

// using the extended Euclidean algorithm.

// (i.e. GCD(a, b) = s.a + t.b).

// O(GCD(a, b)) = O(log(n))

pair<int, int> extendedEuclid(int a, int b) {

if (b == 0) {

return{ 1, 0 };}

pair<int, int> p = extendedEuclid(b, a % b);

int s = p.first;int t = p.second; return{ t, s - t \* (a / b) };}

13) **Pascal Triangle**

int arr[100][100];    
    
for (int line = 0; line < 100; line++)   
{   
  *// Every line has number of integers*   
    *// equal to line number*   
    for (int i = 0; i <= line; i++)   
    {   
    *// First and last values in every row are 1*   
    if (line == i || i == 0)   
        arr[line][i] = 1;   
    *// Other values are sum of values just*   
    *// above and left of above*   
    else   
        arr[line][i] = arr[line-1][i-1] + arr[line-1][i];   
    }   
}

14)inclusion&excusion هام

***//nuber shoud be brime***

ll arr[]={};

ll inc\_exe(int idx = 0, int d = 1, int sign = -1) {

if (idx == عدد العناصر) {

if (d == 1)return 0;

return sign \* n / d;

}

return inc\_exe(idx + 1, d, sign) + inc\_exe(idx + 1, d \* arr[idx], sign\*-1);

}

***15)invers Moduar***

//to make the value positive

1. int md(int val, int md){
2. return (val%md + md) % md;
3. }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. //to solve (a/x)%m?
2. // Modular inverse of the given number modulo m
3. // (a / b) % mod = (a% mod) \* (b%mod) ^ mod - 2
4. // if mod is prime
5. // x^(-1) % mod = power(x,mod -2)
6. // else x^(-1) = inv(x,mod)
8. // Calulate Mod Inv
9. ll inv(ll a, ll m){
10. ll m0 = m, t, q;
11. ll x0 = 0, x1 = 1;
12. if (m == 1)
13. return 0;
14. while (a > 1) {
15. q = a / m;
16. t = m;
17. m = a % m, a = t;
18. t = x0;
19. x0 = x1 - q \* x0;
20. x1 = t;
21. }
22. if (x1 < 0)
23. x1 += m0;
24. return x1;
25. }

28. // Modular inverse of the given number modulo m
30. int modInverse(int a, int m) {
31. return power(a, m - 2);
32. }

***##notes on problem;***

//the num of factor may be odd or even if the num have square rot the factor will be odd else the factor even

//(unsigned int) from 0 to 2^32-1

//(int ) from 0 to 2^31-1

***Big intager***

Int MXN=600;

string s, t;

while (cin >> s >> t) {

reverse(all(s));

reverse(all(t));

vector<int>ans(MXN,0);

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

for(int j =0; j <t.size();j++ ) { //you can write first t.size(),and second s.size()&&make the change in the loop(i,j)

int multipy = ( (s[i] - '0') \* (t[j] - '0') );

ans[i+j] += multipy;

}

}

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

// int cur= ans[i] % 10;

//ans[i+ 1] += ans[i] - cur;//

ans[i+1]+=ans[i]/10;

ans[i] %= 10;

}

// cout << "kkk";

int i = MXN -1;

while (i > 0 && ans[i] == 0)i--;

for (; i>=0;i--) {

cout<<ans[i];

}

cout << "\n";

ans.clear();

}