modulus vs remainder

-3%2=-1 or (-1)*3%2=-1
Manually add x+=m if x<0 since modulus >0;

long long a=1e18,b=2e18;
int m = 1e9+7; // -> 1000000007
int x = (a+b)%m;
cout<<x; //garbage value; since a+b precedence higher.

NUMBER THEORY

Binary exponentiation of x^n in O(logn) using recursion or iteration.

```
long long binpow(long long a, long long b) {
    long long res = 1;
    while (b > 0) {
        if (b & 1)
            res = res * a;
        a = a * a;
        b >>= 1;
    }
    return res;
}


//Caution: If the numbers are small then only use pow()
function, otherwise use the Binary Exponentiation
method to calculate power.
```

Count no. of divisors of a no. in O(root N).

GCD calculation using Euclidean theorem in O(logn)

```
int gcd(int a, int b)
{
    if (a == 0)
        return b;
    return gcd(b % a, a);
}
T.C. O(log(n))
```

LCM=a*b/GCD

Store bool isprime[i] till ‘n’ using Sieve of Eratosthenes

https://www.interviewbit.com/problems/prime-numbers/

```
  vector <bool> isPrime(A+1,1);
   isPrime[0]=isPrime[1]=0;
   for(int i=2;i*i<=A;i++){
       if(isPrime[i]){
           for(int j=i*i;j<=A;j+=i){
               isPrime[j]=0;
           }
       }
   }
```

*//i*i<=A because say a no. i>root(A) ,if it is !prime,its spf will be not be //greater than root(A),because spf*spf will become>A,so it will surely be made 0 //by its spf.*
*//j=i*i because say j=7 so for j=7*2,7*3,...7*6 1,2,3..6 <7 so their multiples //have been made 0 already.*

In time=O(nlog (logsqrt(n))) and space=O(n).

Store spf[i] till ‘n’ using Sieve of Eratosthenes logic

```
for(int i=0;i<=1e6;i++){
spf[i] = i;}

for(int i=2;i*i<=1e6;i++){
if(spf[i]==i){
for(int j=i*i;j<=1e6;j+=i)
if(spf[j]==j) spf[j]=i;
}
}
```

In time=O(nloglogn) and space=O(n)

Sum of prime divisors in O(nloglogn)

While calculating the is_prime[i] array using sieve of eratosthenes,also calculate sum[i].

```
for(int i=2;i<=MAX;i++){ if(is_prime[i]==true){
for(int j=i;j<=MAX;j+=i){ if(j>i) is_prime[j]=false;
sum[j]+=i; } } }
```

—--

//here we will not do i*i<=MAX or j=i*i because we have to add i for every no. j so we can’t skip any i.

q queries of no. of divisors of a no.s <=n:

Brute=O(q*rootn) (optimal for 1 query)

Better-O(qlogn + nloglogn)

Use is_prime and spf array

N=z1^k1 * z2^k2 * z3^k3 * …….(zi-> prime number).
Number of divisors = (k1+1)*(k2+1)*(k3+1)......

```
while(q--){
int n;
cin>>n;
int ans=1; (12)
while(n>1){
int k=0;
int spf = SPF[n];
while(n%spf==0){
n/=spf;
k++;
}
ans=ans*(k+1);
}
cout<<ans<<endl;
}
```

Euler Totient Function:-
O(nloglogn)
phi(n) = count of numbers from 1 to n that are
coprime(gcd(x,y)=1) with n.
//p=prime no.
phi(p) =p-1
phi(p^2) =p^2 - p //since p is prime ,every no. that has p in its factor should be subtracted⇒ p,2*p,3*p,...,p*p

phi(p^k) =p^k - p^(k-1) //since p is prime ,every no. that has p in its factor should be subtracted⇒ p,2*p,3*p,...,p*p,...,p^p ⇒total p^(k-1) no.s
phi(a*b) = phi(a).phi(b) if a and b are coprime,//cram
phi(n)= n*(1-1/p1)*(1-1/p2)....(p1,p2,..prime factors of n) using above

Modulo Inverse

->If x (written as a^-1) is modulo inverse of a w.r.t n then (a*x)%n=1.
->It exists only when gcd(a,n)=1.
->If x=mod inv of a w.r.t. n and y=mod.inv of b w.r.t. a ,then a*x+b*y=1 //no need
->(a^ETF(b))%b =1 (if a&b are coprime).
->Modulo inverse of a w.r.t m(i.e. (a^-1)%m)) is equal to **((a^(m-2))%m;** (iff m is prime number,can be derived from above).
->(a^z)%m = (a^(z%ETF(m)))%m //no need

Three Methods to calculate nCr%m

1.When m is not prime (n,r<10^3)
DP->nCr=(n-1)C(r-1)+(n-1)C(r)

2.Modulo Inverse method
M must be prime and n,r<10^6

3.Lucas Theorem //combiniatrics
M must be prime and n,r can be<10^18.

Important properties
(a-b)%k = (x-y) Then (a-x)%k = (b-y)%k //can be derived by a-b=k*n+(x-y)