Number Theory

Modular Arithmetic

1. (a+b)%M = (a%M + b%M)%M

   (a + b)%M = (q_1*M + r_1 + q_2*M + r_2)%M
   = (M*(q_1 + q_2) + (r_1+r_2))%M
   = (a%M + b%M)%M (Because (M*(q_1 + q_2))%M = 0
   r_1 = a%M
   r_2 = b%M)

2. (a - b)%M = (a%M - b%M + M)%M

3. (a*b)%M = ((a%M) * (b%M))%M

4. (a/b)%M = ((a%M)*(b_inverse%M))%M

   (a/b)%M = ((a%M)*(1/b)%M)%M
   (1/b)%M = b_inverse%M
   b_inverse%M exists only when b and M are coprime i.e. gcd(b,M) = 1

   In order to find b_inverse%M we use Fermat's Little theorem

   Fermat's Little Theorem :

   (a^(p-1))%p = 1

   CONDITIONS
   1. p should be prime
   2. gcd(a,p) = 1

   Example : a = 4
   p = 5
   (4^(5-1))%5 = (256)%5 = 1

   Multiplying with a^(-1) on both sides
   (a^(p-2))%p = (a^(-1))%p = a_inverse%p Eq.(1)

   PS : a^b symbol here refers to a to the power b and not a XOR b

Therefore using Eq(1)
(a/b)%M = ((a%M)*(b_inverse%M))%M = ((a%M)*((b^(M-2))%M))%M

Finding (a^b)%c in O(log(b))

1) Recursive Method

Basic Idea:

a^b = (a^(b/2))*(a^(b/2)) if b is even
a*(a^(b/2))*(a^(b/2)) if b is odd

```
long long int power(long long int ,long long int b,long long int MOD)
{
   if(b==0)
          return 1;
   long long int ans = power(a,b/2);
   ans = (ans*ans)%MOD;
   if(b%2==1)
          ans=(ans*a)%MOD;
   return ans;
}
```

Note : Think why the complexity of the code changes from O(log(b)) to O(b) if we use the following code :

```
long long int power(long long int ,long long int b,long long int MOD)
{
   if(b==0)
          return 1;
   long long int ans = power(a,b/2)*power(a,b/2);
   if(b%2==1)
          ans=(ans*a)%MOD;
   return ans;
}
```

2) Pre-computation method to find power of a number

If we want to compute a^b

here we go on power of 2 and use only those powers of 2 which appear in binary representation of b.

Suppose we want to compute 10^12

Binary representation of 12 = 1100 = 2^3 + 2^2

Therefore 10^12 = 10^(2^3 + 2^2) = 10^(2^3)*10^(2^2)

Hence we precompute an array pow where pow[i] = a^(2^i)

```
long long int pow[30];
pow[0]=a;
for(i=1;i<=30;i++)
   pow[i]=(pow[i-1]*pow[i-1]);  //If required , take mod

//we have to compute a^b
long long int ans=1;
for(i=0;i<=30;i++)
   if((b&(1<<i))>0)
         ans=(ans*pow[i]);
// Here also take mod
```

Euclid algorithm for computation of GCD

```
int gcd(int a,int b)
{
   if(b==0)
        return a;
   return gcd(b,a%b);
}
```

// In built function is also there in c++

__gcd(a,b)

where both a and b should be same either long long int or int.

Sieve of Eratosthenes :

```
#define MAX 100001
int prime[MAX];
If prime[i]=1 then it is prime and if it is 0 then
it is composite.
void sieve(){
   for(i=2;i<MAX;i++)
         prime[i]=1;
   for(int i = 2; i*i<= MAX; i++){
         if(prime[i] == 1){
               for(int j = i*2; j<MAX; j+=i){
                     prime[j] = 0;
               }
         }
   }
}
```

Ques: Count the number of factors of all numbers from 1 to N
1<= N <= 10^6

```
#define MAX 1000000
int factors[MAX+1];  //factors[i] stores the number of factors of number i
int i,j;
for(i=1;i<=MAX;i++)
{
   for(j=i;j<=MAX;j+=i)
         factors[j]++;
}
```