Prime

```
#define MAX 1000001
char prime[MAX]; // 0 দিয়ে initialize করতে হবে
// prime[index] এ যদি 0 থাকে তাহলে index একটি প্রাইম নাম্বার
// আর যদি 1 থাকে তাহলে index প্রাইম না
void primeGenerator( int n ) // n পর্যন্ত প্রাইম বের করব
{
   int x = sqrt( n );
   prime[0] = prime[1] = 1;      // 0 এবং 1 প্রাইম না
   for( int i = 2; i <= x; i++ ) {
      if( prime[i] == 0 ) {     // i যদি মৌলিক সংখ্যা হয় তাহলে 2i
         for( int j = i+i; j <= n; j += i ) // থেকে শুরু করে i
            prime[j] = 1;     // এর সকল গুণিতককে বাদ দিয়ে দিব
      }
   }
   return;
}
```

Prime factorization

```
ll solve(int x){

int h= sqrt(x);
for(ll i=0;i<prime.size() && prime[i]<=h;i++)
{
```

```
If(x%prime[i]==0){

While(x%prime[i]==0){

ll y= x/prime[i];
factor.pb(prime[i];
x=y;
}
}
If(x>1)factor.pb(x);
```

```
prime.push_back(2)
for(int k=3; k<=n; k+=2)
{  if (!on(k))
     { prime.push_back(k);
     }
}

}
```

GCD: 2 টি সংখ্যাকে সবচেয়ে বড় যে সংখ্যা দিয়ে নিঃশেষে ভাগ করা যায়;

~~#define G.~~

ভাগ পদ্ধতি:- 24, 10

```
10) 24 (2
    20
    ---
     4) 10 (2
         8
        ---
        2) 4 (2
           4
          ---
           0
```

~~#define gcd (x,y)~~

```
void gcd ( int x, int y)
{
   return y==0? a : gcd (y, x%y));
}
```

for avoid limit exceed

```
~~void~~ int lcm (int x, int y)
{
    return((~~first~~ x/gcd (x,y)* ~~o~~ y);
}
```

~~#~~ prime factor: ~~[illegible]~~ - prime number

যখন যেকোনো সংখ্যাকে prime number দ্বারা ভাগ যায়, একটি সংখ্যার prime সংখ্যাগুলোর গুণফল আকারে প্রকাশ করাই হলো prime factorization.

184 = 2^k

```
~~for (int i = 0; i*i <= .~~

// solve (int x) {
  int n = sqrt (x)
   for (~~i~~ i = 0;  i < prime.size() &&
                       prime[i] <= n; i++)
      if (x % prime[i] = = 0) {
  {    while (x% prime[i] = = 0)
       {    // y = x/prime[i]
            factor. pb (prime [i]; x = y;
       }
       n = sqrt (x)                     factor. pb(x); }
```

Modular arithmatic

১০০! এর মধ্যে কয়টা ডিজিট আছে? হিসাব করলে দেখা যায় ১৫৮ টির মতো। বলা হলো আপনাকে ১০০! ফাক্টরিয়াল বের করে তার আউটপুট কে ৯৭ দিয়ে ভাগ করে তার ভাগশেষ কে প্রিন্ট করতে হবে। এখন কি আমরা কোনোভাবে অভারফ্লো (Overflow) এড়িয়ে গিয়ে সমধান করতে পারি? ১৫৮ ডিজিটের কোনও সংখ্যাতো ৬৪ বিট আনসাইনড এও ধরবে না। কিন্তু আমরা **মডুলার অ্যারিথমেটিক (Modular arithmetic) এর সূত্র** ব্যবহার করে এই ধরনের সমস্যা সমাধান করতে পারি।

মডুলার অ্যারিথমেটিকে (Modular arithmetic) চলক একটা নির্দিষ্ট সংখ্যায় পোঁছানোর পর আবার ০ থেকে রিপিট হয়। উদাহরণে বুঝা যাক,

একটি কাটা ঘরির কথা ভাবি। যেখানে ঘড়িটি ১ টা থেকে ১২ টা পর্যন্ত সময় দেখাতে পারে। ধরি এখন ৭ টা বাজে। এর ৮ ঘণ্টা পরে ৩ টা বাজবে। আমরা যোগ করে পাই, ৭+৮=১৫, কিন্তু যেহেতু ঘড়িটি প্রত্যেক ১২ ঘণ্টা পরে পরে আবার আগের অবস্থানে আসে, তাই আমাদের ঘড়িতে (৭+৮)%১২ বা ৩ টা বেজেছে। আশা করি বুঝা গিয়েছে কি হচ্ছে। নিজ হাতে কাটা ঘড়ি থাকলে ঘুরিয়ে দেখা যেতে পারে। ☺

বেশিরভাগ প্রোগ্রামিং ল্যাঙ্গুয়েজ (programming language) গুলোতে অপারেটর দিয়ে ভাগশেষ বুঝানো হয়। কে দিয়ে ভাগ করার পর ভাগশেষ প্রোগ্রামিং এ এর মান বের করা।

একে **x** mod **m** পড়া হয়। যেহেতু ১০০! অনেক বড় সংখ্যা, তাই ধরে নেই ১০০!=x। অর্থাৎ x বা ১০০! ফাক্টরিয়াল কে m বা ৯৭ দিয়ে ভাগ করে ভাগশেষ প্রিন্ট করাই আমাদের মুল সমস্যা।

উপরে যা বললাম, আমরা ১০০! বের করতে পারবো না। এটা অনেক বড় সংখ্যা, তবে আমরা ৯৭ দিয়ে ভাগ করে ভাগশেষ বের করতে পারি। এটা int ডাটা টাইপেই ধরে যাবে। যাইহোক, এ ধরনের সমস্যা সমাধানে আমরা নিচের দুটি সূত্রের সাহায্য নিবো। প্রথমে সূত্রগুলোতে চোখবুলাই একটু,

$(a+b)\%m= ((a\%m) + (b\%m))\%m$... ... (১)

$(a*b)\%m= ((a\%m) * (b\%m))\%m$ ... ... (২)

n সংখ্যক সংখ্যা এর জন্য সুত্র দুটি ব্যবহার করতে পারবো। উপরের সমস্যা সমাধানে আমাদের ২য় সূত্রটি কাজে লাগবে। অর্থাৎ সমীকরণের বামপক্ষ ধরে এখন সমাধান করতে হবে। এভাবে করলে আমাদের ওভারফ্লো করবে না। কারণ প্রতিটি ধাপে গুণফলকে ৯৭ দ্বারা mod করা হবে।

নিচের C++ কোডটি দেখি,

C++

```cpp
int big_factorial(int x,int m){
  int fact=1;
  for(int i=1;i<=x;++i){
     fact=((fact%m)*(i%m))%m;
  }
  return fact;
}
```

১০০! এর জন্য এর আউটপুট হবে ০। কারণ ১০০! কে ৯৭ নিঃশেষে ভাগ করে। এখানে দেখা যাচ্ছে আমরা লুপ এর ভিতরে কাজ করেছি দুটি করে সংখ্যা নিয়ে। একটু লক্ষ করলেই আশা করি বুঝা যাবে।

Link : https://iishanto.com/modular-arithmetic-and-modular-inverse-bangla-tutorial/

```
ll bigmod (int x, int power, int mod_value)
{
    if (power = =0) return 1;
    if (power % 2 = =0)
     {
        ll ans= bigmod (x, power/2, mod_value)
        return ((ans % m)*(ans % mod_val)) % mod_val
     }
    if (power % 2 != 1)
     {
        ll ans2 = (x* bigmod (x, power-1, mod_value)) % mod_value
        return ans2;
     }
}
```

$(x^{y^z}) \% m$:—

we have to at first calculat y^z.

here y^z have too many digits. So.

we have ~~to~~ ~~mod~~ calculat $y^z \%$ something

$$x^{y^z} \% m = x^{(y^z \% (m-1))} \% m.$$

Proof :- definition of modulo : $x^y \% m \Rightarrow$,

$y^z \% m = 1$

if m is prime

Divisibility rules

https://flexbooks.ck12.org/cbook/ck-12-cbse-maths-class-6/section/3.5/primary/lesson/divisibility-rules/