→ Modular Multiplicative Inverse (MMI)

→ $(A|B) \,\%\, M \neq \left(\frac{A \,\%\, M}{B \,\%\, M}\right) \%\, M$ ✗

We cannot write it like this.

Instead:

$$(A|B) \,\%\, M = (A \times B^{-1}) \,\%\, M$$
$$= ((A \,\%\, M) \times (B^{-1} \,\%\, M)) \,\%\, M$$

Here, $B^{-1} \,\%\, M$ → MMI of B

→ We know:

When, $A \times B = 1$
Then, B is multiplicative inverse (MI) of A

Similarly,

When, $(A \times B) \% M = 1$

$\Rightarrow (A \% M \times B \% M) \% M = 1$

Then, B is MMI of A. where, $(1 \leq B \leq M-1)$

→ How to find B?

Loop B => 1 → M-1
If $(A \times B) \% M = 1$
Found;

Time complexity : O(M)

NOTE: MMI is defined iff A and M are Coprime (GCD(A, M) == 1).

→ This O(M) time complexity is not good, so to find MMI optimally, we will use Fermet Theorem.

$A^{M-1} \equiv 1 \mod (M)$ → Fermet Theorem

This states that when we divide A^{M-1} with M, (A^{M-1} / M), we will always get 1 as remainder.

Given that:
* M is prime
* A is not multiple of M.

-> Multiply this Fermat Theorem with A^{-1} on both sides.

$\therefore$ $\boxed{A^{M-1} \equiv A^{-1} \bmod (M)}$

↳ This means:

$(A^{M-2} \,\%\, M) = A^{-1}$

We can calculate this optimally using Binary Exponentiation function. (which we have studied earlier). eg: BINEXP(A, M-2).
Time Complexity : O(log(N)).

NOTE: We have taken the case where M is prime, But what if we get M a non-prime number, then we have to find inverse using Extended Euclid Algorithm (EEA), which is not frequently asked. So, we can skip this for now.

-> Question on MMI:

There are N ~~children~~ students and K toffee. K < N. Count the number of ways to distribute toffee among N students, such that each students get 1 toffee only.

Constraints :
$M = 10^9 + 7$
$K < N < 10^6$
$Q < 10^5$

Hint : We have to calculate ^nC_K

$$^nC_K = \frac{n!}{(n-k)! * k!}$$

CODE

```
CONST INT M = 1e9+7;
```

→ To precompute all the values of N, we are precomputing because, we have to calculate our solution for Q queries, and calculating factorial takes O(N) T.C. So, we can't use O(N) complexity in each query.

```
CONST INT N = 1e6+10;
INT FAC[N];
```

→ Binary exponentiation function, (Here, to calculate inverse)

```
INT BINEXP(INT A, INT B);
{
    INT RESULT = 1;
    WHILE (B > 0)
    {
        IF (B & 1)
        {
            RESULT = (RESULT * 1LL * A) % M;
```

```
        }
        A = (A * 1LL * A) % M;
        B >>= 1;
    }
    RETURN RESULT;
}

INT MAIN()
{
    FACT[0] = 1;

    -> Computing factorials
    FOR (INT i = 1; i < N; i++)
    {
        FACT[i] = (FACT[i-1] * 1LL * i) % M;
    }

    -> Queries
    INT Q;
    CIN >> Q;
    WHILE (Q--)
    {
        INT n, K;
        CIN >> n >> K;
        -> Numerator
        INT ANS = FACT[n];
        -> Denominator
        INT DEN = (FACT[n-K] * 1LL * FACT[K]) % M;
        -> Inverse of denominator, because,
        -> (A/B) % M = ((A%M) * (B^-1 % M)) % M
        ANS = ANS * BINEXP(DEN, M-2);
```

```
        cout << ANS;
        cout << ANS * BINEXP(DEN-
        cout << ANS;
    }
    RETURN 0;
}
```