cannot be
pobl.

```
for (char& ch : s) {
    cout << ch << "\n";
    ch = 'z';
}
cout << s << "\n";
```

Output :-
zzzz

So this can be used for modify char.

```
for (int i=0; i<5; i++) {
    cout << i << "\n";
}
```

Output :-
0
1
2
3
4

This is the classic for loop

Modular Arithmetic Foundation

Why Modulo is used ?
→ To avoid overflow
→ To keep numbers in range.

Exa : const int MOD = $10^9 + 7$
int result = (a+b) % MOD

Why this is used ?
1. Prime
2. If It fits in a 32-bit integer, It is less than $\frac{INT\_MAX}{2}$. So no overflow
3. Tradition
4. First prime after 10^9.

Rule of Modulo :-

① Use long long

$$Exp = ((A+B) - (C \times D) + (E^F)) \% M$$

$$\downarrow$$

$$[(\quad) \pm * \div (\quad)] \% M$$

$$[((\quad) \% M) \ \text{Operation} \ ((\quad) \% M)] \% M$$

Exa :- using lli = long long int; OR #define int int long long

~~Exa :-~~ void solve(){

    int a, b, c, d, e, f, m;
    cin >> a >> b >> c >> d >> e >> f >> m;
    int temp1 = (a+b) %m;
    int temp2 = ~~(c+d) % m~~; ((c %m) * (d%m)) %m [if

}

② ~~Modulo with subtraction~~ While printing :

We need to handle negative numbers while printing by adding '+m'.

$$(a-b)\%m = ((a\%m) - (b\%m) + m)\%m;$$

Range : $-\infty \le x < \infty \xrightarrow{\%M}$

$$-(M-1) \le x\%M \le (M-1) \xrightarrow{+M}$$

$$1 \le (x\%M)+M \le 2M-1 \xrightarrow{\%M}$$

$$0 \le (x\%M+M)\%M \le (M-1)$$

③ ~~Use~~ binpow instead of pow ~~for exponential.~~

~~int temp4 = (temp3 + binpow (e, f, m)) %m~~

④ For negative numbers.

Range : $[0, m-1]$.

Exa: int mod (int a, int m){

    return ((a%m)+m)%m;

}

④ <u>Calculation of exponents</u> :-

For $a^x \xrightarrow[\text{steps}]{\text{two}}$ we can go $a^{x/2}$

odd → even → half

even → half

Exa:-

$2^{38} \to 2^{14} \to 2^7 \to 2^6 \to 2^3 \to 2^2 \to 2^1$

If odd, doing 1 the step to even

$\boxed{A^b \% M} \to$ binpow (a, b, M) $\xrightarrow{\text{odd}}$ $a \cdot$ binpow (a, b-1, M) % M

binpow (a, b, M) $\xrightarrow{\text{even}}$ (binpow (a, b/2, M))^2 % M

B is odd → A^{B-1} % M

B is even → $(A^{B/2})^2$ % M → $(A^{(B/2)}$ % m$)^2$ % m

<u>Code</u> :-

```
int binpow (int a, int b, int m) {
    if (b==0) return 1;
    if (b%2) return (a*b binpow (a, b-1, m))%m;
    else {
        int temp = binpow (a, b/2, m);
        return temp * temp %m ;
    }
```

⑤ <u>For division (÷)</u>

Use inverse

$\left(\frac{P}{Q}\right)$ % M ⇒ (P%M) × (Q%M)$^{-1}$ % M

~~For Exa:-~~

<u>Explanation of inverse</u> :-

3^{-1} % M $\simeq \frac{1}{3}$ % 7

3×3^{-1} % 7 = 1 % 7

$3 \cdot 3^{-1}$ % 7 = 1

∴ For, $\boxed{a \cdot a^{-1} \% m = 1}$

$\boxed{\text{For } a^{-1} \text{ exist} \to \gcd(a, m) == 1}$

∴ This rule works only when :
→ m is prime
→ Q is not divisible by m

Exa : $a \cdot a^{-1}$ % 7 = 1

a	0	1	2	3	4	5	6
a^{-1}	X	1	4	5	2	3	6

To find inverse of any number with modulo m and assumes p is prime no.

code:
```
int inv(int x, int p){
    return binpow(x, p-2, p);

void solve(){
    int temp5 = temp4 * inv(g) % m;
}
```

Rules :-

① 2 at a time : (a * b) % m ⇒ (a%m * b%m) % M
(take mode if needed)

② Use long long ⇒ #define int long long

③ When you are printing don't directly : cout << ans;
Use : cout << (ans % m + m) % m . (Handle -ve output)

④ Use binpow (a, b, m)

⑤ If $\left[\frac{A}{B} \% M\right]$ → M will be prime
→ $A \cdot B \% M$
→ $A \cdot B^{M-2} \% M \Rightarrow (A \% M) \cdot ((B^{M-2}) \% M) \% M$