

```

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

```

Output: -
z z z z

→ So this can be used for modify char.

```

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

```

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: $\text{const int MOD} = 10^9 + 7$
 $\text{int result} = (a+b) \% \text{MOD}$

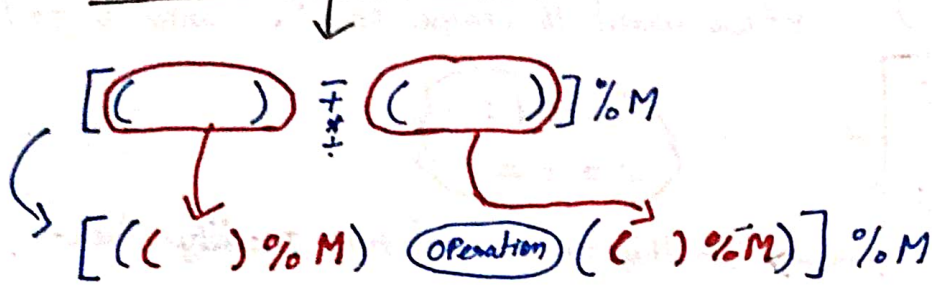
Why this is used?

- ① Prime
- ② If it fits in a 32-bit integer, it is less than $\frac{\text{INT_MAX}}{2}$. So no overflow
- ③ Tradition
- ④ First prime after 10^9 .

Rule of Modulo :-

① Use long long

$$\text{Exp} = \frac{(A+B) - (C \times D) + (E^F)}{M}$$



Exa:-

using `lli = long long int;` OR `#define 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'.

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

$$\text{Range: } -\infty \leq x < \infty \rightarrow \% M$$

$$\rightarrow -(M-1) \leq x \% M \leq (M-1) \rightarrow +M$$

$$1 \leq (x \% M) + M \leq 2M-1$$

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

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

$$\text{int temp4} = (\text{temp3} + \text{binpow}(e, f, m)) \% m$$

④ For negative numbers.

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

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

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

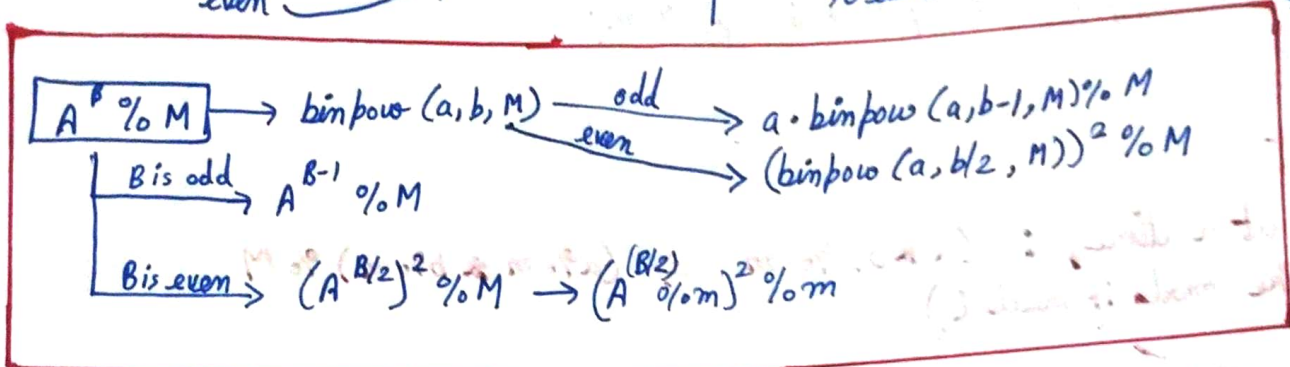
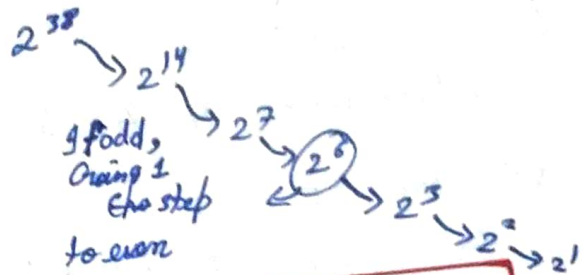
`}`

④ Calculation of exponents :-

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

odd \rightarrow even \rightarrow half
even \rightarrow half

Ex:-



Code :- `int binpow (int a, int b, int m) {`

```

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

⑤ For division (\div)

Use inverse

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

For Ex:-

Explanation of inverse :-

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

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

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

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

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

a	0	1	2	3	4	5	6
a ⁻¹	X	1	4	5	2	3	6

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

\therefore This rule works only when :
 $\rightarrow m$ is prime
 $\rightarrow a$ is not divisible by m

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, : $(a * b) \% m \Rightarrow (a \% m * b \% m) \% m$
(take mod if needed)
- ② Use long long \Rightarrow # 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 $\left[\frac{A}{B} \% M\right] \rightarrow M$ will be prime
 $\rightarrow A \cdot B^{-1} \% M$
 $\rightarrow A \cdot B^{M-2} \% M \Rightarrow (A \% M) \cdot (B^{M-2} \% M) \% M$