* Bit-wise operator :

→ AND (&)
→ OR (|)
→ XOR (^)
→ One's Complement (~)
→ Left Shift (<<)
→ Right shift (>>)

1→ Binary And (&)

```
1 & 0 = 0
1 & 1 = 1          Ex. 5 & 7 =  101
0 & 1 = 0                     & 111
0 & 0 = 0                     -----
                                101 = 5
```

2→ Binary OR (|)

```
0 | 0 = 0
0 | 1 = 1          Ex → 5 | 8 →  0101
1 | 0 = 1                        1000
1 | 1 = 1                        ----
                                 1101 = 13
```

3→ Binary XOR (^) → Exclusive OR

```
0 ^ 0 = 0
1 ^ 1 = 0          Ex → 5 ^ 7 → 101
0 ^ 1 = 1                       111
1 ^ 0 = 1                       ---
                                010 = 2
```

4 → Binary Not (~)

~0 = 1 ~5 = ~(000101)

~1 = 0 = 111010

int a = 0

cout << (~a); → this print (-1)

Now In a 32 bits are 0 and In (~a) all bits are 1.

$$\sim a = \overset{\uparrow}{1}\,\underbrace{1 \ldots\ldots 1}_{31 \text{ numbers}}$$

32 bits

sign bit

1 → -ve no.

2s compliment form

so we invert and add 1

Now → $\underbrace{000 \ldots}_{30 \text{ bits}}\ 1 = -1$

5 → Binary left shift (<<)

Now → 5 << 2

$(101)_2 << 2$

$= (10100)_2$

$= 20$

$\boxed{a << b = a \times 2^b}$

6 → Binary Right shift (>>)

$$20 >> 2 = \frac{20}{2^2} = 5$$

$$= (10100)_2 >> 2$$

$$\boxed{= 5}$$

$$\boxed{a >> b = \frac{a}{2^b}}$$

→ Trick → All odd number last bit is 1 and In even number last bit is 0. if (num & 1) == 1 number is odd else number is even.