Truth table of Bitwise Operators

a	b	a&b	a|b	a^b	~a
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Basic And Properties

1) Even / odd number

10 → 1 0 1 0
2^3 2^2 2^1

9 → 1 0 0 1

In binary representation, if a number is even, then its least significant bit (LSB) is 0.

Conversely, if a number is odd, then its LSB is 1.

1) odd / Even Condn.

A&1 → A → x y z a b c
& 1 → 0 0 0 0 0 1
0 0 0 0 0

A&1 → 1 (A is odd)
→ 0 (A is even)

```
if ((A&1) == 1) {   // odd number.
}
```

2) $A \& 0 = \underline{0}$.

$$\begin{array}{lll} & A & \longrightarrow x\ y\ z\ \alpha\ \beta\ \gamma \\ \& & 0 & \ 0\ 0\ 0\ 0\ 0\ 0 \\ \hline & & \ 0\ 0\ 0\ 0\ 0\ 0 \\ \hline \end{array}$$

3) $A \& A \longrightarrow A$

$$\begin{array}{lll} & A \rightarrow & 1\ 0\ 1\ 1\ 0 \\ \& & A \rightarrow & 1\ 0\ 1\ 1\ 0 \\ \hline & A \underline{\rightarrow} . & 1\ 0\ 1\ 1\ 0 \\ \hline \end{array}$$

OR properties

1) $A | 0 = A$

$$\begin{array}{lll} & A \rightarrow & 1\ 0\ 1\ 1\ 1 \\ OR & 0 \rightarrow & 0\ 0\ 0\ 0\ 0 \\ \hline & & 1\ 0\ 1\ 1\ 1 \\ \hline \end{array}$$

2) $A | A = A$

$$\begin{array}{ll} A \rightarrow & 1\ 0\ 1\ 1\ 0 \\ A \rightarrow & 1\ 0\ 1\ 1\ 0 \\ \hline & 1\ 0\ 1\ 1\ 0 \\ \hline \end{array}$$

XOR Properties

1) $A \wedge 0 = A$

```
     A            A
     ↓            ↓
  —  1  —      —  0  —
     0            0
  ---------    -------
     1            0
  ---------
```

```
        A →  1 0 1 1 1
  XOR   0 → 0 0 0 0 0
            ----------
             1 0 1 1 1
            ----------
```

2) $A \wedge A = \underline{0}$.

```
        A                 A
        ↓                 ↓
  ———   0   ———     ———   1   ———
  ———   0   ———     ———   1   ———
  ---------------   ---------------
        0                 0
                    -------------
```

```
  A →  1 0 1 1 0
  A →  1 0 1 1 0
      ------------
       0 0 0 0 0
      ------------
```

Cummulative Property

↳ Order doesn't change the result.

a & b = b & a

a | b = b | a

a ^ b = b ^ a

Associative Property

↳ grouping doesn't impact the overall result.

(A & B) & C = A & (B & C)

(A | B) | C = A | (B | C)

(A ^ B) ^ C = A ^ (B ^ C).

Ques

Evaluate the expression: a ^ b ^ a ^ d ^ b

$a \wedge b \wedge a \wedge d \wedge b$

↓

$\underline{a \wedge a} \wedge b \wedge b \wedge d$

↓

$\underline{0 \wedge b} \wedge b \wedge d$

↓

$\underline{b \wedge b} \wedge d$

↓

$\underline{0 \wedge d}$

↓

d

Ques

Evaluate the expression: 1 ^ 3 ^ 5 ^ 3 ^ 2 ^ 1 ^ 5

$1 \wedge 3 \wedge 5 \wedge 3 \wedge 2 \wedge 1 \wedge 5$

↓

$\underline{1 \wedge 1} \wedge \underline{3 \wedge 3} \wedge \underline{5 \wedge 5} \wedge \underline{2}$

↓

$0 \wedge 0 \wedge 0 \wedge 2$

↓

2

Left Shift Operator (<<)

Let's say we have 8 bit numbers,

$a = \underline{10}$

		7	6	5	4	3	2	1	0	
a = 10	=	0	0	0	0	1	0	1	0	=> 10
a << 1	=	0	0	0	1	0	1	0	0	=> 20
a << 2	=	0	0	1	0	1	0	0	0	=> 40
a << 3	=	0	1	0	1	0	0	0	0	=> 80
a << 4	=	1	0	1	0	0	0	0	0	=> 160
a << 5	=	0	1	0	0	0	0	0	0	=> 64

waste

waste

gone

64 ↓ overflow, significant bit got lost .

$$\boxed{\begin{array}{l} a << n = a * 2^n \\ 1 << n = 2^n \end{array}}$$ (assuming no overflow)

Right Shift Operator (>>)

		7	6	5	4	3	2	1	0	
a = 20	=>	0	0	0	1	0	1	0	0	→ discarded
a >> 1	=>	0	0	0	0	1	0	1	0	=> 10
a >> 2	=>	0	0	0	0	0	1	0	1	=> 5
a >> 3	=>	0	0	0	0	0	0	1	0	=> 2
a >> 4	=>	0	0	0	0	0	0	0	1	=> 1
a >> 5	=>	0	0	0	0	0	0	0	0	=> 0.

$$\boxed{\begin{array}{l} a >> n = \dfrac{a}{2^n} \\[1em] 1 >> n = \dfrac{1}{2^n} \end{array}}$$

* Set ith bit

1<<4

		5	4	3	2	1	0
	N=	1	0	1	1	0	1
OR	(1<<4)	0	1	0	0	0	0
		1	1	1	1	0	1

		5	4	3	2	1	0
	N=	1	0	1	1	0	1
OR	1<<3	0	0	1	0	0	0
		1	0	1	1	0	1

To set ith bit of a number

$N = N \mid (1<<i)$

Toggle ith bit

		5	4	3	2	1	0
	N=	1	0	1	1	0	1
XOR	(1<<4)	0	1	0	0	0	0
		1	1	1	1	0	1

		5	4	3	2	1	0
	N=	1	0	1	1	0	1
XOR	1<<3	0	0	1	0	0	0
		1	0	0	1	0	1

$N = N \wedge (1<<i)$

check bit at particular idx.

```
                5 4 3 2 1 0
        N=      1 0 1 1 0 1
And   (1<<4)    0 1 0 0 0 0
      -----------------------
                0 0 0 0 0 0   → 0
```

```
                5 4 3 2 1 0
        N=      1 0 1 1 0 1
AND    1<<3     0 0 1 0 0 0
      -----------------------
                0 0 1 0 0 0   → non zero.
```

```
if ( N & (1<<i)) == 0) {
    // ith bit in n was unset
}
else {
    // ith bit was set.
}
```

1<<3

```
N=12 →   1 1 0 0
       & 1 0 0 0
       ---------
         1 0 0 0
```

```
0 0 0 0 0 0 0 1    (1)
0 0 0 0 0 0 1 0    (1<<1)
0 0 0 0 0 1 0 0    (1<<2)
```

```
                      N=12    3
function checkBit ( N, i ) {
    if ( N & (1<<i)) == 0) {
        return false;
    }
    else {
        return True;
    }
}
```

T.C → O(1)

S.C → O(1)

```
            7 6 5 4 3 2 1 0
N =         0 0 1 0 1 1 0 1

            7 6 5 4 3 1 0
N<<3        1 1 0 1 000
```

	7	6	5	4	3	2	1	0
1 →	0	0	0	0	0	0	0	1
(1<<3)	0	0	0	0	1	0	0	0

Ques   Count no. of set bits in N.

N = 12 . ⟶ 1100 → Ans → 2 .

Approach 1 :-

```
                        N=12
function countBit (N) {
    ans = 0;                                   T.C -> O(1)
    for (i=0; i<32; i++) {                     S.C -> O(1) .
        if (checkBit (N, i)) {
            ans++
        }
    }
    return ans;
}
```

Approach 2 :-

N = 1010 ⟶ 8bit no

```
N =        0 0 0 0 1 0 1 0
&          0 0 0 0 0 0 0 1
           ---------------
           0 0 0 0 0 0 0 0

N>>1 =>    0 0 0 0 0 1 0 1
&    =>    0 0 0 0 0 0 0 1
           ---------------
           0 ----------- 0 1
```

```
n >> 2
   & 1 ,        0  0  0  0  0  0  1  0
                0  0  0  0  0  0  0  1
              ___________________________
                0 ——————————————— 0  0
              ___________________________

n >> 3 ,        0  0  0  0  0  0  0  1
                0 ——————————————— 0  1
              ___________________________
                0 ——————————————— 0  1
              ___________________________

n >> 4  →    0  0  0  0  0  0  0  0  →  0 .
```

```
ans = 0;                              T.C → O(logn)
                                      S.C → O(1)
while (n>0) {
    if ((n&1)!=0)
        ans++;
    }
    n=(n>>1)
}
```

Ques unset ith bit of a no' n.

N = 6, → $\overset{2}{1}\ \overset{1}{1}\ \overset{0}{0}$ i = 2

↓

Ans 0 1 0 → 2.

① check ith bit.

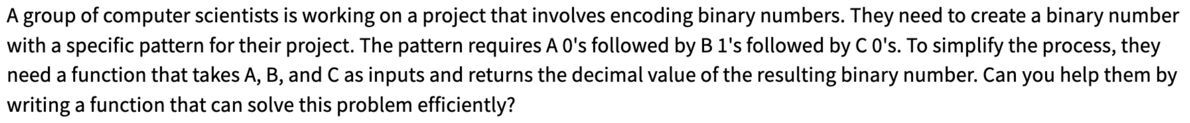
② if bit is set then do XOR.

```
func unset(N,i){
      if (checkBit(N,i)){
            N = N ^ (1<<i)
      }
      return N;
}
```

T.C → O(1)

S.C → O(1).

Ques Set Bits in a Range .



A group of computer scientists is working on a project that involves encoding binary numbers. They need to create a binary number with a specific pattern for their project. The pattern requires A 0's followed by B 1's followed by C 0's. To simplify the process, they need a function that takes A, B, and C as inputs and returns the decimal value of the resulting binary number. Can you help them by writing a function that can solve this problem efficiently?

A = 4, B = 3, C = 2

→ 0 0 0 0 1 1 1 0 0 ← 28 Ans.

e.g

A = 4, B = 3, C = 2

7 6 5 4 3 2 1 0

N = 0   0 0 0 0 0 0 0 0

A = 2, B = 4, C = 3

7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0

```
ans = 0;
for (i = 0; i < B; i++) {   → 4.   0,1,2,3
    setBit(N, C+i);
}
```

N = N | (1<<i)