Today's Agenda :-

Hey everyone, Welcome ☺

Announcement :-

13th October - Contest date

3A, 2B } 5 sessions { Arrays ✓, Bit Manipulations ✓

Start Preparing !!

Bitwise Operators & , | , ^ , ~ , << , >>

Truth Table :-

for the res to be 1 both bits should be 1

either of the bit is set, the res is 1

same same puppy shame

a	b	&	|	^ → abs(a-b)
0	0	0	0	0
✓ 0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

1's complement

~1 = 0

~0 = 1

Properties :-

If a no is even, the LSB is 0

Conversly, for odd no, the LSB is 1

18

10010 → 1

21

10101

$A \& 1 = 1$

↳ A is odd

$A \& 1 = 0$

↳ A is even

```
   A          LSB
   ---------------
&  0 0 0 0 0  1
   ---------------
   0 0 0 0 0  1/0
```

Eg 181 =

```
   10110101
&  100000001
   ----------
   00000001
```

Eg 180 =

```
   10110100
&  100000001
   ----------
   00000000
```

$$\boxed{A \,\&\, 0 = 0}$$

$$\boxed{A \,\&\, A = A}$$

&

$$\begin{array}{r} 1\,0\,0\,1\,0\,1\,0 \\ 1\,0\,0\,1\,0\,1\,0 \\ \hline 1\,0\,0\,1\,0\,1\,0 \end{array}$$

$$\boxed{A \mid 0 = A}$$

$$\boxed{A \mid A = A}$$

|

$$\begin{array}{r} 1\,0\,0\,1\,0\,1\,0 \\ 1\;0\,0\,0\,0\,0\,0\,0 \\ \hline 1\,0\,0\,1\,0\,1\,0 \end{array}$$

$$\boxed{A \wedge 0 = A}$$

$$\boxed{A \wedge A = 0}$$

^

$$\begin{array}{l} A = 1\,0\,0\,1\,0\,1\,0 \\ \wedge\;\;0\,0\,0\,0\,0\,0\,0 \\ \hline A = 1\,0\,0\,1\,0\,1\,0 \end{array}$$

$$\begin{array}{r} 1\,0\,1\,0\,0 \\ 1\,0\,1\,0\,0 \\ \hline 0\,0\,0\,0\,0 \end{array}$$

Commutative :-

$A \& B = B \& A$

$A \mid B = B \mid A$

$A \wedge B = B \wedge A$

$1 \& 0 = 0$

$0 \& 1 = 0$

$$\begin{array}{r} 10101 \\ 01101 \\ \hline \end{array}$$

Associative :-

$(A \& B) \& C = A \& (B \& C)$

$(A \& C) \& B$

$(A \mid B) \mid C = A \mid (B \mid C)$

$(A \wedge B) \wedge C = A \wedge (B \wedge C)$

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

$\rightarrow (a \wedge a) \wedge (b \wedge b) \wedge d$

$0 \wedge 0 \wedge d$

$0 \wedge d$

d ✓

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

$\rightarrow (1 \wedge 1) \wedge (3 \wedge 3) \wedge (5 \wedge 5) \wedge 2$

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

$0 \wedge 0 \wedge 2$

$0 \wedge 2$

2

Risk of overflow

$$\boxed{a << 1 = a * 2}$$

Left Shift (<<) → shifts the bits of a no. to left by specified no. of positions.

$a = 10$:	0 0 0 0 1 0 1 0	→ 10	
			$*2$
$a << 1$:	0 0 0 1 0 1 0 0	→ 20	$(10 * 2^1)$
			$*2$
$a << 2$:	0 0 1 0 1 0 0 0	→ 40	$(10 * 2^2)$
			$*2$
$a << 3$:	0 1 0 1 0 0 0 0	→ 80	$(10 * 2^3)$
	7 6 5 4 3 2 1 0		$-2^7 + 2^5 = -128 + 32$
$a << 4$:	1 0 1 0 0 0 0 0	→ −96	$(10 * 2^4)$ 160
$\boxed{a << 5}$:	0 1 0 0 0 0 0 0	→ 64	ov (10×2^5) 320

Overflow

8 bits : [−128, 127]

overflow

$$\boxed{a << N = a * 2^N}$$ → assuming there's no overflow

Put $a = 1$

$$\boxed{1 << N = 2^N}$$

Right Shift (>>) → shift the bits towards right by specified no of pos.

$a = 20$	0 0 0 1 0 1 0 0	→ 20
	↘ ↘ ↘ ↘ ↘ ↘ ↘ ↘	]/2
$a >> 1$:	0 0 0 0 1 0 1 0	→ 10 $\left(\frac{20}{2}\right)$
	↘ ↘ ↘ ↘ ↘ ↘ ↘ ↘	]/2
$a >> 2$:	0 0 0 0 0 1 0 1	→ 5 $\left(\frac{20}{2^2}\right)$
	↘ ↘ ↘	
$a >> 3$:	0 0 0 0 0 0 1 0	→ 2 $\left(20/2^3\right)$
	↘ ↘	
$a >> 4$:	0 0 0 0 0 0 0 1	→ 1 $\left(20/2^4\right)$
	↘	
$a >> 5$:	0 0 0 0 0 0 0 0	→ 0 $\left(20/2^5\right) = 0$

$$\boxed{a >> N = a/2^N}$$

8 bit

Eg $1 << \underline{3}$: 0 0 0 0 0 0 0 1

1

1

0 0 0 0 1 0 0 0 } $2^3 : \underline{8}$

3 2 1 0

$a << N = a * 2^N$

$1 << 3 = 1 * 2^3 = \underline{8}$

$1 << N = 2^N$

$1 << 3 = 2^3$ }

Set i^{th} bit of a no.

Make it 1

	5	4	3	2	1	0	
N	1	0	1	1	0	1	→ Set 4th bit
1<<4 →	0	1	0	0	0	0	
	1	1	1	1	0	1	

2^4 : 1<<4

	5	4	3	2	1	0	
N	1	0	1	1	0	1	→ Set 3rd bit
1<<3 →	0	0	1	0	0	0	
	1	0	1	1	0	1	

$$\boxed{N = N \mid (1 << i)}$$

2^i

→ Set i^{th} bit of N.

$$\boxed{N = N \wedge (1 << i)}$$

2^i

Toggle | flip i^{th} bit of a no.

	5	4	3	2	1	0
	1	0	1	1	0	1
^	0	0	0	1	0	0
	1	0	1	0	0	1

	5	4	3	2	1	0
	1	0	1	1	0	1
^	0	1	0	0	0	0
	1	1	1	1	0	1

Q. Check if i^{th} bit of a no is set or not

	5	4	3	2	1	0
	1	0	1	1	0	1
&	0	0	1	0	0	0
	0	0	1	0	0	0

if (N & 1 = 1)

N is odd

	0	1	0	1	0	0	1
&	0	0	0	0	0	0	1
	0	0	0	0	0	0	1

$i = 1$

$1 << 1$

$\hookrightarrow 2$

	5	4	3	2	1	0
	1	0	1	1	0	1
&	0	0	0	0	1	0
	0	0	0	0	0	0

```
if (N & (1 << i) == 0) {
    // ith bit of N is unset
}
else {
    // ith bit of N is set
}
```

≥ 0

N

&

2^i ← 0 0 0 1 0 0 0 0

$\parallel$

$1 << i$

0 0 0 0 0 0 0 0

i^{th}

0

2^5

5	4	3	2	1	0
1	0	0	0	0	0

H/W Try using right shift operator (Discussed in Intermediate)

$2^i \rightarrow$ only

$\boxed{N = 45,\ i = 2}$ $\longrightarrow$

	5	4	3	2	1	0
	1	0	1	1	0	1
&	0	0	0	1	0	0
	0	0	0	1	0	0
	5	4	3	2	1	0

$= 2^2$

```
if ( N & (1 << i) > 0) {
    |        Set
    }
else
    |        unset
    }
```

```
bool checkBit (N, i) {
    if (N & (1<<i) == 0) {
        return false
    }
    else {
        return true
    }
}
```

Q Given N, count total no of set bits in N

N = 12 : 1100 } ans = 2.

integer : 32 bits

↳ 4 Bytes : 32 bits

↳ [0, 31]

```
int countSetBits (int N){
    int cnt = 0
    for (int i = 0; i < 32; i++){
        if ( checkBit (N, i)){
            cnt = cnt + 1
        }
    }
    return cnt
}
```

TC : O(1)

32 iteration

Approach 2 :-

int

..0 0 . . . . 0 0 0 0 1 0 1 0 1 0 0 1

Cnt = 1

0 1 0 1 0 1 0 0

$N >> 1 = \frac{N}{2^1}$

```
func(    ) {
    int cnt = 0
    while (N > 0) {
        if(N & 1 == 1) {
            cnt = cnt + 1
        }
        N = N >> 1 // N = N/2
    }
    return cnt
}
```

$O(\log_2 N)$

Q Unset ith bit of if it is set

$$\boxed{N \,\&\, 0 = 0}$$

```
        3 2   1 0
        1 1   0 0
      ^
1<<2    0 1   0 0
        ---------
        1 0   0 0
```

$$\boxed{N \,\hat{}\, (1 << i)}$$

toggles

```
        3 2   1 0
        1 1   0 0
      ^
1<<1    0 0   1 0
        ---------
        1 1   0 0
```

Set → |

toggle → ^

```
Unsetith (int N, int i) {
    if (checkBit(N, i)) {
        N = N ^ (1 << i)
    }
    return N
}
```

→ TC : O(1)
SC : O(1)

Q)

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?

Constraints:

```
0 <= A, B, C <= 20
```

60 bits long is required

8 bytes = 64 bits

$A = 0,\ B = 20,\ C = 20$

20 19 20 0

20 0000 1111

39

32

$2^{32} - 1$

$A = 20,\ B = 20,\ C = 20$

: 60 bits

A 1's B 0's C 1's

40 bits

$A = 4,\ B = 3,\ C = 2$

8 7 6 5 4 3 2 1 0

0 0 0 0 1 1 1 0 0 → 28

0's B 1's C 0's

C^{th} $C-1$ 0

$[B+C-1]$

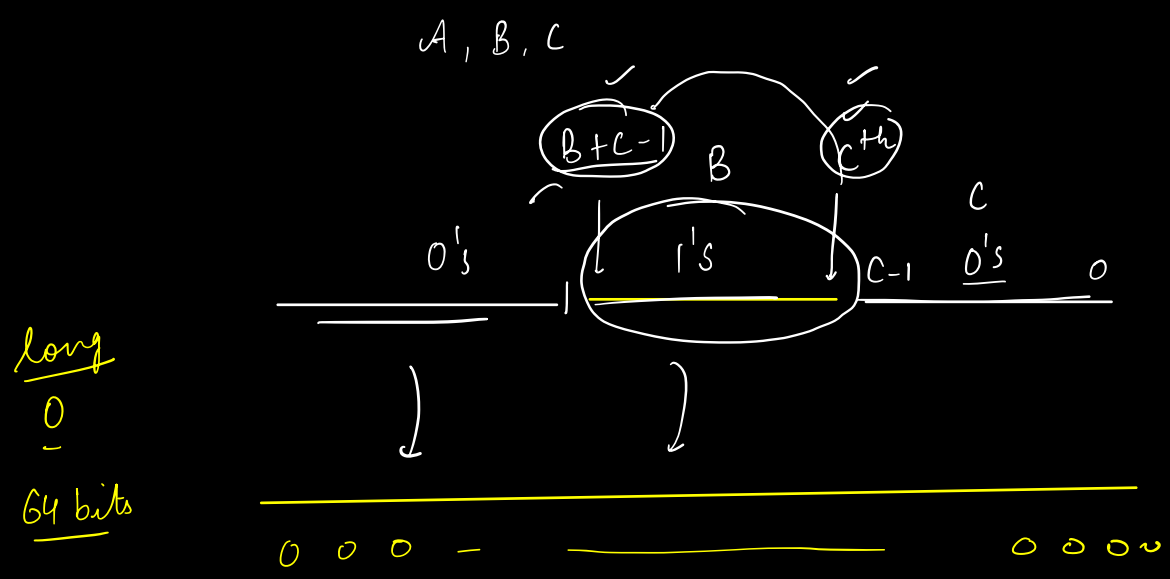
10 9 8 7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0 0 0 0

1 1 1

0 → Set bits from [C^{th} to $B+C-1^{th}$ bit]

```
long ans = 0
                                  i <= B+C-1
for(int i = C ; i < B+C ; i++) {
|     ans = ans | (1<<i)
}
return ans
```



Friday - Bit M-2 (interview Problems) } imp

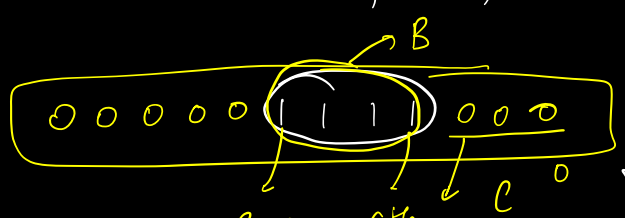
Thank You ☺

Go & revise intermediate Notes.

A, B, C

0's 1's 0's

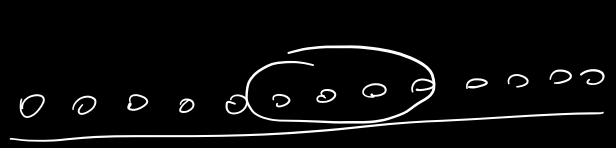
A = 5, B = 4, C = 3



long

0

64 bits



$C - x + 1 = B$

$x = B + C - 1$