$\log_2 n + 1$ → max bits in n
eg for long long int = 10^5
Page No. 64 bits
Date :

Bit Manipulation

(*) And gate

1 & 1 = 1

1 & 0 = 0

0 & 1 = 0

0 & 0 = 0

(*) OR

1 | 1 = 1

1 | 0 = 1

0 | 1 = 1

0 | 0 = 0

eg a & b

ie 5 & 7

```
5 → 0 0 0 0 1 0 1
7 → 0 0 0 0 1 1 1
-----------------
    0 0 0 0 1 0 1 → 5 Ans
```

ie 5 | 6 ⇒

```
  1 0 1
  1 1 0
-------
  1 1 1 → 7 Ans
```

(*) XOR → exclusive OR

1 ^ 1 = 0

1 ^ 0 = 1 → Exactly 1 bit is 1

0 ^ 1 = 1

0 ^ 0 = 0

Another prop. of XOR :

XOR of same no. is 0

i.e. $5 \wedge 5 = 0$

i.e. $5 \wedge 7 \wedge 5 = 7$

$0 \wedge 7 = 7$

(*) ~ Bitwise NOT → flip all bits

$\sim 1 = 0$

$\sim 0 = 1$

(*) Shift operators

i) << left shift
ii) >> Right shift

] also Binary operators (means two operands to operate.)

i) left shift → shift 1 position ~~forward~~ toward left

eg $5 << 1 \Rightarrow$ 0 0 0 1 0 1 = 5

0 0 1 0 1 0 = 10

Thus, left 1 means

⇒ multiply by 2

i.e. $a << 1 \Rightarrow a * 2$

Page No.
Date :

eg: $a << 2 \Rightarrow a * 2 * 2$

i.e. $5 << 2 \Rightarrow 5 * 2 * 2 = 20$

Thus, $\boxed{a << b \Rightarrow a * 2^b}$

Similarly

Right shift (division)

i.e. $\boxed{a >> b \Rightarrow a / 2^b}$

Set = 1

* application

1) to find unique no. in array, where every no. repeat twice.
→ xor of same no. is 0. $(a \wedge b \wedge a) = b$

2) $(n \& 1) = 1 \Rightarrow n$ is odd
$0 \Rightarrow n$ is even

3) find the ith bit (fromRight)

$(N \& (1 << i)) > 0 \; ? \; 1 : 0;$

4) set the ith bit

int mask = $(1 << i)$

$n = (n \mid mask)$

Page No.
Date :

5) clear ith bit

0000 [1] 0 1
~(0000100)
1111011
&. = Ans

① int mask = ~(1<<i)
② n = (n & mask)

6) update ith bit
① clear the ith bit
② take OR with v

i → ith bit
v → update with v (1 or 0)

eg
1[1]01
① clear → 1001
② or 0v00
1v01 //

to clear
int = mask ~(1<<i)
then take &

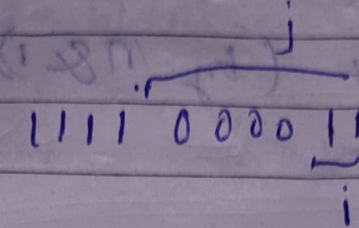
6) clear last i bits.

(-1) = 11111111
or
(~0)

So,
mask = (-1<<i)
N = (N & mask)

0101101 1010
& 11111110000
Ans// 8 = 10110100000

7) clear j to i = Bits.



to get this 2 steps.

① 1111 0000 00 → (-1 << j)

② 0000 0000 11 → $(2^i - 1)$

⇒ ((1 << i) - 1)

OR

mask = 1111 0000 11

Now, simply take & with.

N = N & mask;

8) count no. of set bits in n

```
ans=0
while (n>0) {
    n = (n & (n-1));
    ans++;
}
```

Removes the last set bit (from RIGHT)

OR 2nd way

__builtin_popcount(n)

9) print the binary form.

Step-1 extract the last bit (by (n & 1))

Step-2 to print.

1101

ans * =
=1 * 1
* 0 * 10
+ 1 * 100
+ 1 * 1000

p = 1

p = 10

p = 100

So a = 1101

Code:

```
int ans = 0;
int p = 1;

while (n>0) {
    int last_bit = (n & 1);
    n >> 1;
    ans += last_bit * p;
    p *= 10;
}
cout << ans;
```