Q Given an array of size (N-1) having all elements from 1 to N except one. Find the missing no.

[1 - N]

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 3 | 6 | 1 | 4 | 2 | 7 |

→ 5

[1 - N]

$$\frac{N(N+1)}{2} - \sum_{i=0}^{N-1} A[i]$$

TC: O(N)
SC: O(1)

| 0 | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| 3 | 6 | 1 | 4 | 2 | 7 | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

ans → take XOR of array
take XOR of all elements from 1 to N

A ^ A = 0
A ^ 0 = A

Q Given an array with all elements appearing 2 times except 2 elements.
Return the 2 single elements. (O(1) SC)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 15 | 20 | 30 | 15 | 4 | 20 | 30 | 9 |

→ [4, 9]

15 ^ 20 ^ 30 ^ 15 ^ 4 ^ 20 ^ 30 ^ 9 ⇒ 4 ^ 9

```
  0100
  1001
 ------
  1101   ⇒ 13
 ------
```

1 1 0 1

4 9

**Step I** Take XOR of array — O(N)

XOR = $S_1 \wedge S_2$

**Step II** Find position of any set bit in XOR → (P) — N & 1

```
int getSetBitPos (int xor){
    for(i=0; i<32; i++){
        if(checkBit(xor, i)){
            ret i;
        }
    }
    ret -1;
}
```

O(log Max)

**Step III** Take xor of all elements which have
a set Bit at p in ans1
an unset bit at p in ans2

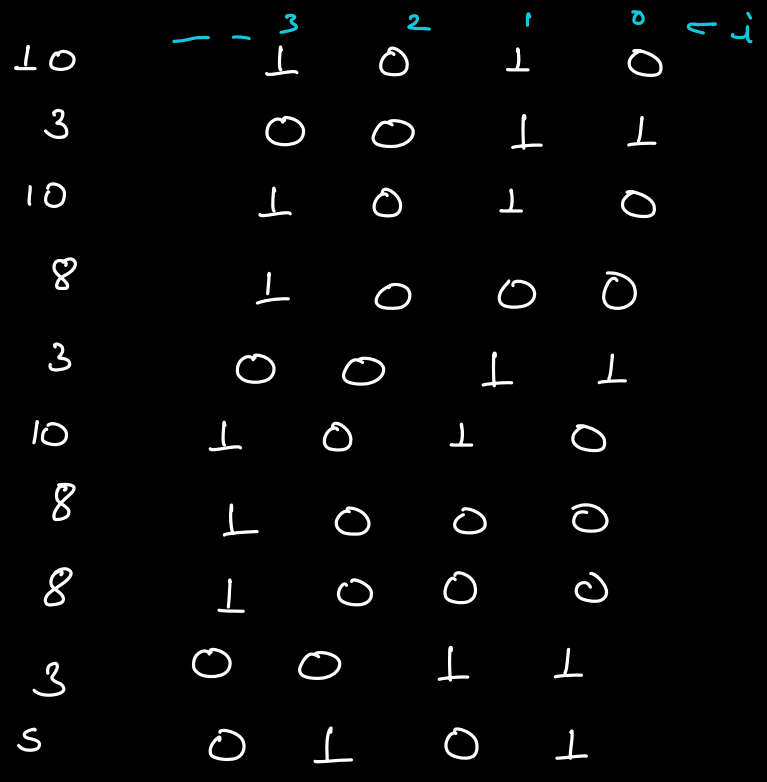
```
ans1 = 0;  ans2 = 0;                                   O(N)
for ( i = 0; i < N; i + + ){
        if( checkBit ( A[i], p)){
                ans1 = ans1 ^ A[i];
        }
        else {
                ans2 = ans2 ^ A[i];
        }
}
```

TC : O(N)
SC : O(1)

Q Given an array where all no. appear 3 times except 1.
Find the single no. (O(1) SC)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 3 | 10 | 5 | 8 | 3 | 10 | 8 | 8 | 3 |

⟶ 5




0 0 0 0 0 0 0 ... 0

```
ans = 0;
for(i=0; i<32; i++){                    ⟶ O(log M)
    count = 0;
    for(j=0; j<N; j++){
        if(checkBit(A[j], i)){
            count++;
        }
    }
    if(count % 3 != 0){
        ans = setBit(ans, i);
    }
}
return ans;
```

return (ans | (1<<i))

TC: O(N log Max)

SC: O(1)