Todays Content :

- >> Right Shift
- checkBit()
- countSetBits()
- SetBits()
- -Ve are stored

>> Right Shift :

8bit number $2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$a = 10$: 0 0 0 0 1 0 1 0 $\longrightarrow 2^3 + 2^1 = 10$

add → discard

a>>1 : 0 0 0 0 0 1 0 1 $\longrightarrow 2^2 + 2^0 = 5 = 10/2$

discard

a>>2 : 0 0 0 0 0 0 1 0 $\longrightarrow 2^1 = 2 = 10/2^2 = \frac{10}{4}$

discard

a>>3 : 0 0 0 0 0 0 0 1 $\longrightarrow 2^0 = 1 = 10/2^3 = \frac{10}{8}$

discard

a>>4 : 0 0 0 0 0 0 0 0 $\longrightarrow = 0 = 10/2^4 = \frac{10}{16}$

Observations :

a>>1 : $a/2$

a>>2 : $a/2^2$

a>>3 : $a/2^3$

Note: We don't have to worry about overflow in right shift

a>>n : $a/2^n$

// Size:

			Bit position
int	4 Bytes	32 bit	[0 31]
long	8 Bytes	64 bit	[0 63]

1 Byte = 8 bits

// Problems:

checkBit(N, i) : Given Integer N, check if i^{th} bit in N is set (1) or Not (0)

Ex 1:

N = 21 : 1 0 1 0 1 (positions 2^4 2^3 2^2 2^1 2^0)

i = 2

Set → {return True}

N = 34 : 1 0 0 0 1 0 (positions 2^5 2^4 2^3 2^2 2^1 2^0)

i = 3

Unset → {return False}

Idea1: Convert N → binary & simply check if i^{th} bit pos is Set or Not?

Ex3: N = 82

i^{th} bit

	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
N	1	0	1	0	0	1	0	0 → N&1==1 → Set else → Unset
N>>1	–	1	0	1	0	0	1	1 → (N>>1)&1==1 → Set Else → Unset
N>>2	–	–	1	0	1	0	0	2 → (N>>2)&1==1 → Set Else → Unset
N>>3	–	–	–	1	0	1	0	3 → (N>>3)&1==1 → Set Else → Unset
N>>4	–	–	–	–	1	0	1	4 → (N>>4)&1==1 → Set Else → Unset

Given N, check i^{th} (N>>i)&1==1 → Set else → Unset

```
boolean checkBit(int N, int i) {        TC: O(1)  SC: O(1)
    if((N>>i)&1 == 1) {
        // ith bit in N is set
        return True
    }
    else { return false }
}
```

TODO → Try it with &2 operator

Q: Given a Integer N, count no: of set bits in N

N >= 0 → 4 Bytes → 32 bits → [0 31]

Ex1: N = 10 : 1 0 1 0 → 2 set bits

Ex2: N = 27 : 1 1 0 1 1 → 4 set bits

Idea1: Convert N → binary & get no: of 1s in Binary

Idea2: Iterate from [0-31] and at every bit position, check if, the particular bit position is set or Not

```
int countbits(int N){
    int c = 0;
    i=0; i<32; i++){
        // in N, check if ith bit is set or Not
        if( N>>i&1 == 1) {c=c+1}
    }
    return c;
}
```

Idea3:

N = 45 :, C = 0

2^5	2^4	2^3	2^2	2^1	2^0	N&1 == 1 : C = C+1
1	0	1	1	0	1	C = C+1, C = 1
0	1	0	1	1	0	C = 1
0	0	1	0	1	1	C = C+1, C = 2
0	0	0	1	0	1	C = C+1, C = 3
0	0	0	0	1	0	C = 3
0	0	0	0	0	1	C = C+1, C = 4
0	0	0	0	0	0	N==0, break

```
int countSet(int N){
    int c=0
    while(N>0){
        if(N&1==1){
            c=c+1
        }
        N = N>>1
    }
    return c;
}
```

We need to update N

Note:

N = 10 → N = 10

print(N+10) : 20 print(N>>1) : 5

print(N) : 10 print(N) : 10

3Qs) Given N, i, set i^{th} bit in N, & return updated N → N is integer $0 <= i <= 31$

If i^{th} bit N is unset → set

If i^{th} bit N is set → leave it

Ex1: N = 10 : 1 0 1 0 (positions $2^3\ 2^2\ 2^1\ 2^0$)

i = 2 (bit 1)

N = 1 1 1 0 → 14

N = 23 : 1 0 1 1 1 (positions 4 3 2 1 0)

i = 2 : leave it

N = 1 0 1 1 1 → 23

Idea1: Using if/else

```
int setBit(int N, int i){        // i ∈ [0, 31]
    if ( (N >> i) & 1 == 0 ){
        // ith bit in N is unset
        // If we set ith bit value will increase by 2^i
        N = N + 2^i          Note: (1 << i) = 2^i
        return N;
    }
    else{
        return N;
    }
}
```

N = N >> i

N = N & 1

Ex2:

	2^5	2^4	2^3	2^2	2^1	2^0	
N=38	1	0	0	1	1	0	→ N
i=3	0	0	1	0	0	0	→ 2^3
$N/2^3$ =	1	0	1	1	1	0	

	2^5	2^4	2^3	2^2	2^1	2^0	
N=38	1	0	0	1	1	0	→ N
i=2	0	0	0	1	0	0	→ 2^2
$N/2^2$ =	1	0	0	1	1	0	

// generalized : N, i

```
int setbit(N, i) {
    N = N | (1<<i)
    return N
}
```

Note: $1<<i = 2^i$

// doubt:

N=2, i=2 : Set i^{th} bit in N

2^2 2^1 2^0

N=2 : 0 1 0

i=2 : 1 0 0

N|(1<<2) : 1 1 0 ⇒ 6

2^{31} 2^{30} 2^{29} ..... 2^6 2^5 2^4 2^3 2^2 2^1 2^0

N=10: 0 0 0 ..... 0 0 0 1 0 1 0

i=6 0 0 0 ..... 1 0 0 0 0 0 ⇒ 2^6

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

// Given x, y, { Don't worry about overflows }

Create number, such that x continous 1's followed y continous 0's

x	y		output
2	3	11000	24
3	2	11100	28
1	4	10000	16
3	4	1110000	112
2	4	110000	48
3	3	111000 ($2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$)	56

→ TODO: Using loop

- No loop /
- No if else /
- Single line

Hint: Write code to set x Continous 1's?

Ex:

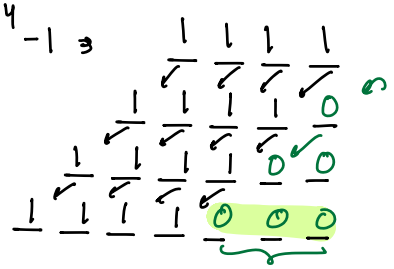
a	$a-1$ → Binary	
2^1	$1 : 2^1-1$	1
2^2	$3 : 2^2-1$	11
2^3	$7 : 2^3-1$	111
2^4	$15 : 2^4-1$	1111
2^5	$31 : 2^5-1$	11111
2^6	$63 : 2^6-1$	111111
2^n-1		11..1 (x continous)

ans = (1<<x)-1

x continous 1's

(1<<n)-1 ⟶ 1's

Ex: x = 4, y = 3 → $2^4 - 1$ ⇒



left shifty by 3 time to get 3 zeros at back

// Given x & y, x continous is followed by y continous 0s

```
int SetNy(int x, int y)
{
    return [(1<<x)-1]<<y;   or   1<<(x+y) - 1<<y
}
```

x continous 1's — y continous 0's

Final:

$[(1<<x)-1]<<y$ $\quad a<<n \Rightarrow a*2^n$

$[2^x - 1]<<y \longrightarrow a*2^y \longrightarrow [2^x - 1][2^y]$

$a = 2^x - 1$

$[2^x - 1][2^y] \longrightarrow 2^x * 2^y - 2^y \rightarrow 2^{x+y} - 2^y$

$[2^{x+y} - 2^y] \longrightarrow 1<<(x+y) - 1<<y$

Avg time to solve this Question {2hrs}