CHECK BIT

Q1 Given two positive numbers N & i. Check if i^{th} bit in N is set or unset.

Set ⇒ the bit is 1 [ON]
Unset ⇒ the bit is 0 [OFF]

Ex1 N ⇒ 10, i = 2

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

⇒ Unset

Ex2 N ⇒ 10, i = 3

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

⇒ Set.

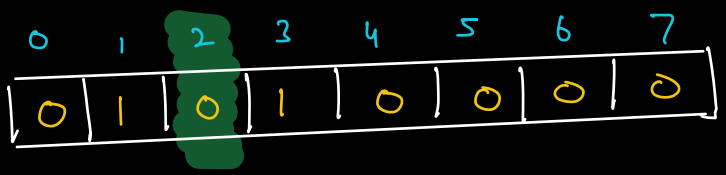
Approach 1

$\Rightarrow$ 7 6 5 4 3 2 1 0

0 0 0 0 1 0 1 0

N = 10

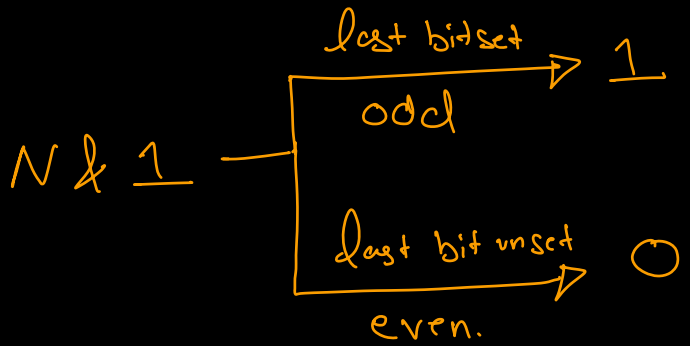
arr[] =



Check arr[i]

Observation !

Check if the rightmost Bit is set or unset.



Approach 2 :

$N \rightarrow 10$, $i = 2$

	7	6	5	4	3	2	1	0
	0	0	0	0	1	0	1	0
$N >> 1$	0	0	0	0	0	1	0	1
$N >> 2$	0	0	0	0	0	0	1	0

$\Rightarrow (N >> i) \& 1$ — set → 1, unset → 0

Pseudo Code

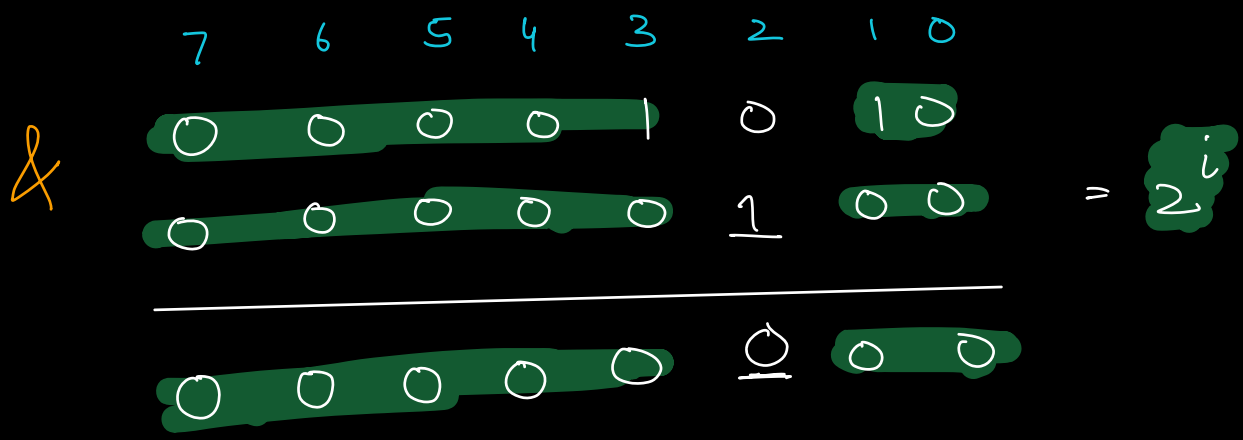
```
bool checkBit(int n, int i) {
    if ((n >> i) & 1 == 1)
        return true;
    else
        return false;
}
```

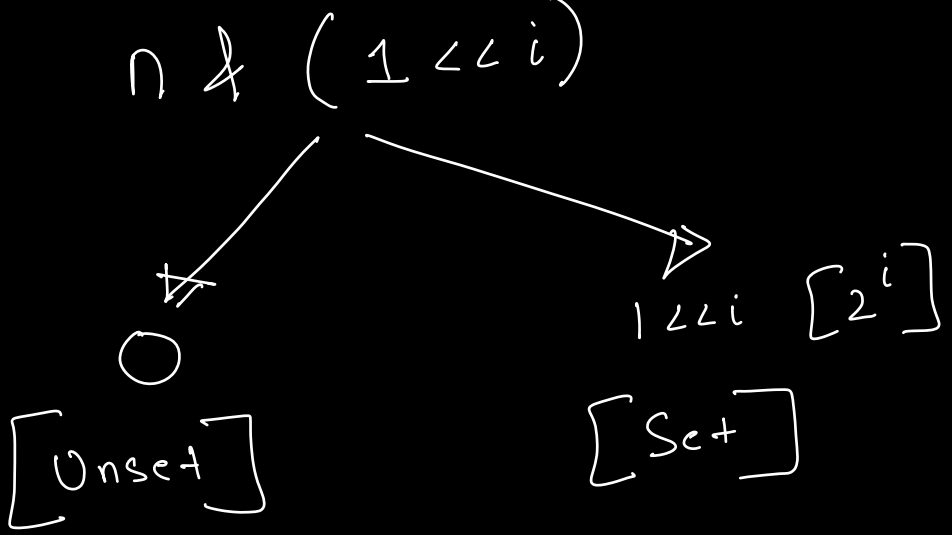
Tc : O(1)

Sc : O(1)

Approach 3

$n = 10$, $i = 2$






```
if (n & (1<<i) != 0)
    return true
else
    return false;
```

SET BIT

Q_2 Given 2 positive numbers N & i. Set the i^{th} bit in N. If i^{th} is 0, change it to 1. If i^{th} bit is 1, let it be.

Ex1 $N = 4 \Rightarrow 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0$

$i = 3 \Rightarrow 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0$

$\Rightarrow 12$

Ex2 $N = 4 \Rightarrow 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0$

$i = 2 \Rightarrow 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0$

$\Rightarrow 4.$

OR

0 | 1 ⇒ 1

1 | 1 ⇒ 1

0 | 0 ⇒ 0

1 | 0 ⇒ 1

Pseudo Code :

```
int setBit (int n, int i) {
    return (n | (1 << i));
}
```

Ex n ⇒ 10, i = 4. = 26

```
  0 0 0 0 1 0 1 0
  0 0 0 1 0 0 0 0
  ---------------
  0 0 0 1 1 0 1 0
  ---------------
```

AND

0 | 1 ⇒ 0

0 | 0 ⇒ 0

```
  0 0 0 0 1 0 1 0
  1 1 1 0 1 1 1 1
  ---------------
  ---------------
```

n ⇒ 10 ⇒ 0 0 0 0 1 0 1 0

1) i = 4 ⇒

```
  0 0 0 0 1 0 1 0
+ 0 0 0 1 0 0 0 0
  ---------------
  0 0 0 1 1 0 1 0
  ---------------
```

$0 \wedge 1 = 1$

$B \wedge 0 = B$

$x_1\ x_2\ x_3\ x_4\ 0\ x_5\ x_6\ x_7$

$0\ 0\ 0\ 0\ 1\ 0\ 0\ 0$

$x_1\ x_2\ x_3\ x_4\ 1\ x_5\ x_6\ x_7$

$x_1\ x_2\ x_3\ x_4\ 1\ x_5\ x_6\ x_7$

$0\ 0\ 0\ 0\ 1\ 0\ 0\ 0$

$x_1\ x_2\ x_3\ x_4\ 0\ x_5\ x_6\ x_7$

```
int setBit (int n, int i) {
    if (!checkBit (n, i)) {
        return n ^ [1 << i]
    }
    else {
        return n
    }
}
```

TC : O(1)

SC : O(1)

n

Q3 Count all set Bits. Given a positive number n. Count all bits which are set.

[Google, Amazon]

Ex1 $n = 10 \Rightarrow 1010 \Rightarrow ans = 2$

Ex2 $n = 8 \Rightarrow 1000 \Rightarrow ans = 1$

Approach 1 :

Input is in integer. !

```
int c = 0
for (int i=0; i<32; i++)
    if (checkBit(n, i))
        c++;
}
return c;
```

Tc: O(32) ≃ O(1)

DATA TYPE DEPENDENT

n => 26

n =>	0 0 0 1 1 0 1 0	c = 0
n >> 1	0 0 0 0 1 1 0 1	c = 1
n >> 2	0 0 0 0 0 1 1 0	c = 1
n >> 3	0 0 0 0 0 0 1 1	c = 2
n >> 4	0 0 0 0 0 0 0 1	c = 3
n >> 5	0 0 0 0 0 0 0 0	

n = 2 = 2
n = 16 = 4
n = 1024 = 10

Pseudo code !

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

Tc: O(log n)

Sc: O(1)

Overflows

Q Add all elements of an array.

$1 \le n \le 10^5$

$-10^6 \le arr[i] \le 10^6$

int : $[-2 \times 10^9, 2 \times 10^9]$

```
int sum = 0
for (int i=0 ; i<n ; i++) {
    sum = sum + arr[i];
}
return sum;
```

Max Value of Sum $\Rightarrow 10^5 \times 10^6$

$\Rightarrow 10^{11}$

Q) Given 2 numbers return $a \times b$.

int a = 10^5

int b = 10^6

$[-2 \times 10^{18}, 2 \times 10^{18}]$

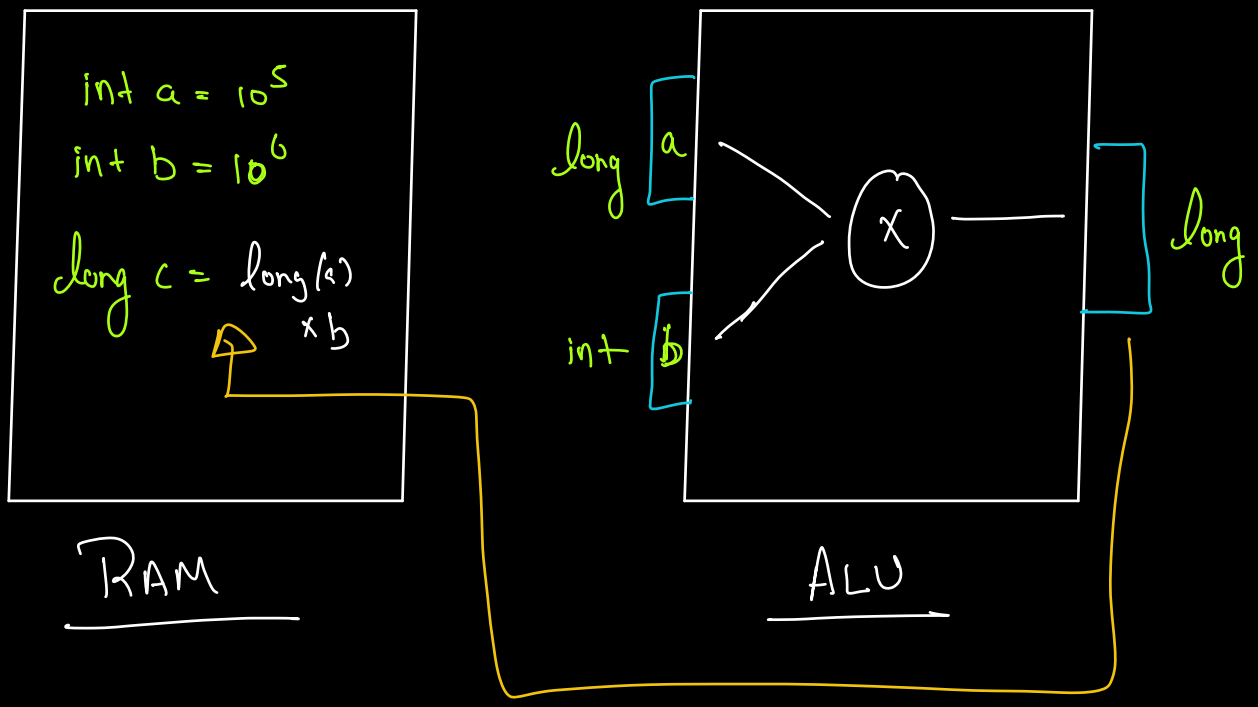
1) int c = a x b ⇒ OVERFLOW

2) long c = a x b ⇒ OVERFLOW

3) long c = long (a x b) ⇒ OVERFLOW

4) long c = long (a) x b ⇒ Will work

5) long c = long (a) x long (b) ⇒ will work



int to str type casting = O(1) ?

int c => a x b x c

Negative numbers in Binary

Byte x = 10 : 0 0 0 0 1 0 1 0