8 bit number

Power of left shift

	7	6	5	4	3	2	1	0	
N = 45 →	0	0	1	0	1	1	0	1	
(1 << 2) →	0	0	0	0	0	1	0	0	
OR	0	0	1	0	1	1	0	1	→ 45

	7	6	5	4	3	2	1	0	
45 →	0	0	1	0	1	1	0	1	
(1 << 4) →	0	0	0	1	0	0	0	0	
	0	0	1	1	1	1	0	1	→ 61

$45 + 2^4$

$\boxed{N \mid (1 << i)}$ → N ⇒ i^{th} bit is already set (1)
→ $N + 2^i$ ⇒ i^{th} bit is unset (0)
} i^{th} bit becomes 1.

	7	6	5	4	3	2	1	0	
N = 45 →	0	0	1	0	1	1	0	1	
(1 << 2) →	0	0	0	0	0	1	0	0	
XOR	0	0	1	0	1	0	0	1	→ 41

$45 - 2^2$

	7	6	5	4	3	2	1	0	
45 →	0	0	1	0	1	1	0	1	
(1 << 4) →	0	0	0	1	0	0	0	0	
	0	0	1	1	1	1	0	1	→ 61

$45 + 2^4$

$\boxed{N \wedge (1 << i)}$ → $N - 2^i$ ⇒ i^{th} bit is set (1)
→ $N + 2^i$ ⇒ i^{th} bit is unset (0)
} Toggle i^{th} bit.

	7	6	5	4	3	2	1	0	
N = 45 →	0	0	1	0	1	1	0	1	
(1 << 2) →	0	0	0	0	0	1	0	0	
AND →	0	0	0	0	0	1	0	0	→ 4

2^i

	7	6	5	4	3	2	1	0	
45 →	0	0	1	0	1	1	0	1	
(1 << 4) →	0	0	0	1	0	0	0	0	
	0	0	0	0	0	0	0	0	→ 0

$\boxed{N \,\&\, (1 << i)}$ → 2^i ⇒ i^{th} bit is set
→ 0 ⇒ i^{th} bit is unset

Given an integer N.

	7	6	5	4	3	2	1	0
N = 12 →	0	0	0	0	1	1	0	0

$i = 5$

0 0 1 0 1 1 0 0

1) Set i^{th} bit of N → $\underline{N = N \mid (1 << i)}$
no change if it is already set.

2) toggle i^{th} bit of N → $N = N \wedge (1 << i)$

3) Unset i^{th} bit of N → if(checkBit(N, i))
no change if it is already unset.
$N = N \wedge (1 << i)$
else // no change

4) check if i^{th} bit is set →
$N \mid (1 << i) == N$ ⇒ i^{th} bit is set
$N \wedge (1 << i) < N$ ⇒ i^{th} bit is set
$N \& (1 << i) == 2^i$ ⇒ i^{th} bit is set

Q→ check if i^{th} bit is set in N without left shift operator.

7 6 5 4 3 2 1 0
N=45 → 0 0 1 0 1 1 0 1

i = 3 → Ans = true
i = 4 → Ans = false

N & 1
→ 1 → 0^{th} bit is 1
→ 0 → 0^{th} bit is 0

7 6 5 4 3 2 1 0
N=45 → 0 0 1 0 1 1 0 1

N >> 3 → 0 0 0 0 0 1 0 1

$(N >> i) \& 1$
→ 1 ⇒ i^{th} bit is set
→ 0 ⇒ i^{th} bit is unset

Q→ Given an integer N, count no. of set bits in N.

N = 45 → 0 0 1 0 1 1 0 1 Ans = $\underline{4}$

N = 10 → 0 0 0 0 1 0 1 ⓪   Ans = $\underline{2}$

N = 10
N = N >> 1
print(N)

N = 10
N = N + 2
print(N) → 12

N = 10 → 0 0 0 0 1 0 1 ⓪   Ans = $\underline{2}$

N >> 1   0 0 0 0 0 1 0 ①

N >> 2   0 0 0 0 0 0 1 ⓪

N >> 3   0 0 0 0 0 0 0 1

N >> 4   0 0 0 0 0 0 0 0 → $\underline{\text{stop}}$

N & 1 == 1 ✓

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

ans += (N & 1)

$TC = O(\log_2(N))$

$SC = \underline{O(1)}$

$\log_2(2^{32}) = \underline{32}$

2^{32}

int → 32

long → 64

$\boxed{10:30 \text{ PM}}$

---

8 bit number

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

[0 — 255]

$$2^7 + 2^6 + 2^5 + \dots 2^0 = \frac{2^0 * (2^8 - 1)}{2 - 1} = 2^8 - 1 = \underline{255}$$

Negative Numbers (2's complement)

MSB → most significant bit

signed bit

(-ve)

1 → -ve

0 → +ve

$$2^7 > (2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0)$$

$$\frac{2^0(2^7 - 1)}{2 - 1} = \underline{2^7 - 1}$$

N = ㊺ →

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

flip all bits →	1	1	0	1	0	0	1	0	→ 1's complement
add 1 → +	0	0	0	0	0	0	0	1	
-45 ←	1	1	0	1	0	0	1	1	→ 2's complement

$$(-2^7) + 2^6 + 2^4 + 2^1 + 2^0 = 211$$

$$-2^7 + (2^6 + 2^4 + 2^1 + 2^0) = -128 + (64 + 16 + 2 + 1)$$

$$= -128 + 83 = \underline{-45}$$

$(-12) \rightarrow (?)_2$

	7	6	5	4	3	2	1	0
12 →	0	0	0	0	1	1	0	0
						1	1	
flip →	1	1	1	1	0	0	1	1
add 1 →	0	0	0	0	0	0	0	1
	1	1	1	1	0	1	0	0

$$-2^7 + 2^6 + 2^5 + 2^4 + 2^2 = -128 + (64 + 32 + 16 + 4)$$

$$= -128 + 116 = \underline{-12}$$

45 - 12

45 + (-12)

	7	6	5	4	3	2	1	0
discard ← ①	1	1	1	1	1			
45 →	0	0	1	0	1	1	0	1
-12 →	1	1	1	1	0	1	0	0
	0	0	1	0	0	0	0	1

$$2^5 + 2^0 = 32 + 1 = \underline{33}$$

$$\boxed{45 - 12 = 33}$$ ✓

Ranges

8 bit system

	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
	7	6	5	4	3	2	1	0	
min →	1	0	0	0	0	0	0	0	$\rightarrow -2^7 = \underline{-128}$
max →	0	1	1	1	1	1	1	1	$\rightarrow 2^7 - 1 = \underline{127}$

$\boxed{\text{Range } -128 \text{ to } 127}$

Integers

32 bits

31 30 29 28 - - - 0

min → 1 0 0 0 - - - 0 → $-2^{31} = -2147483648 \approx -2*10^9$

max → 0 1 1 1 - - - 1 → $2^{31}-1 = 2147483647 \approx 2*10^9$

long

64 bits

63 62 61 - -- 0

min → 1 0 0 - - - 0 → $-2^{63} \approx -9*10^{18}$

max → 0 1 1 - - - 1 → $2^{63}-1 \approx 9*10^{18}$

Q → Given a integer array,
find the sum of all array elements.

```
~~int~~ long ans = 0
for i ⟶ 0 to (N-1)
    ans += A[i]
return ans
```

Constraints

$1 <= N <= 10^5$

$1 <= A[i] <= 10^6$

max total sum = $\underbrace{10^6 + 10^6 + 10^6 --- + 10^6}_{10^5 \text{ times}}$

$= 10^6 * 10^5 = 10^{11}$ ✓

constraints → datatypes
→ TLE

Q → Find a*b for given integers a & b.

max a → $2*10^9$

max b → $2*10^9$

max a*b → $4*10^{18}$ → long

```
int ans = a*b          ✗
return ans
    ↓
long ans = a*b         ✗
return ans
    ⇓
long ans = long(a*b)   ✗
return ans
    ||
```

overflow at multiplication ✓

int * int → int

⇓
long ans = long(a) * b ✓

return ans

long * int → long ✓

long ans = a

ans *= b ✓

return ans

long ans = (1L * a) * b ✓

return ans

long ans = $\underline{a * 1L} * b$

return ans ✓