$2^2 \longrightarrow 1 << 2$

0 0 0 1

3 2 1 0

0 1 0 0 → 2^2 ✓

3^2 → ✗ $1 << 3$ ✗

2^3 → ✓ $1 << 3$ ✓

N = 45 : 1 0 1 1 0 1

OR

5 4 3 2 1 0

1 0 1 1 0 1 : N

OR

$(1 << 2)$ 0 0 0 1 0 0

1 0 1 1 0 1 : N

5 4 3 2 1 0

OR 1 0 1 1 0 1 N

$(1 << 4)$ 0 1 0 0 0 0

1 1 1 1 0 1 $N+2^4$

$N \mid (1 << i)$

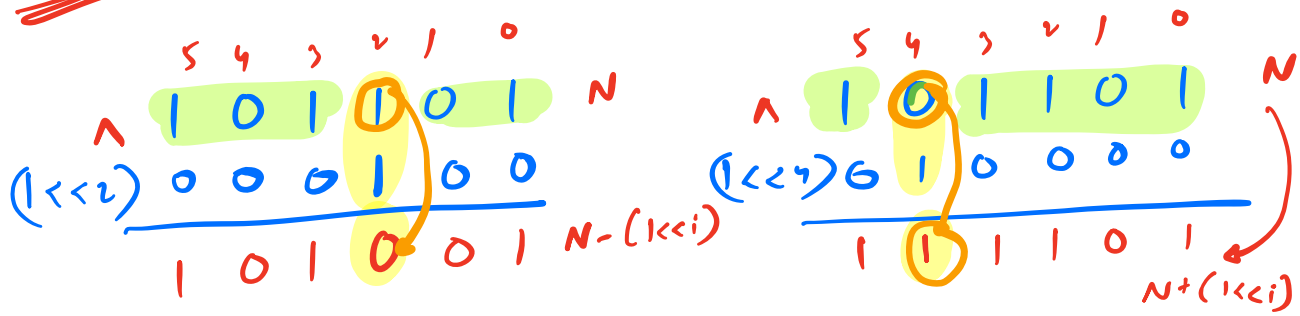
→ N → ith bit in N was set

→ $N + (1 << i)$ → ith bit in N was unset

Q → Given N, set it's ith bit!

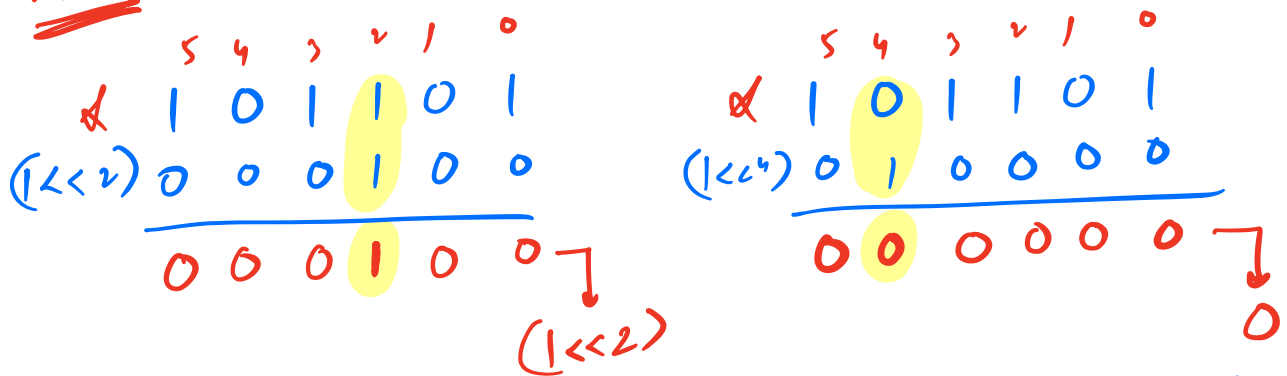
$$N = N \mid (1 << i)$$

XOR



$N \wedge (1 << i)$ → Flip/Toggle the i^{th} bit!

AND



$N \,\&\, (1 << i)$ →
- $(1 << i)$: i^{th} bit is set in N
- 0 : i^{th} bit is unset in N

Q Given N, unset it's ith bit if it's set!

```
if((N & (1<<i)) > 0){ → ith bit is set!
    N = N ^ (1<<i);
}
```


```
  5 4 3 2 1 0
  1 0 1 1 0 1
& 1 1 0 1 1 1   → ~(1<<i)
-------------
  1 0 0 1 0 1
```

N = N & (~(1<<i))

Q Given N, check if the ith bit is set or not?

```
  5 4 3 2 1 0
  1 0 1 1 0 1
& 0 0 1 0 0 0
-------------
  0 0 1 0 0 0
```

(N & (1<<i)) == (1<<i)

```
  5 4 3 2 1 0
  1 0 1 1 0 1
& 0 1 0 0 0 0
-------------
  0 0 0 0 0 0
```

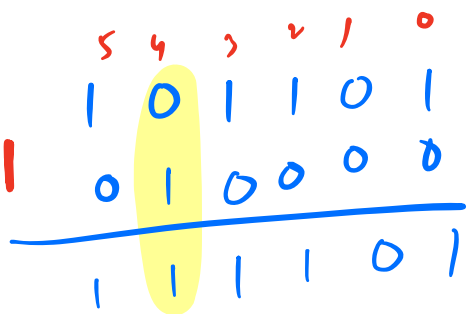
(N & (1<<i)) == 0

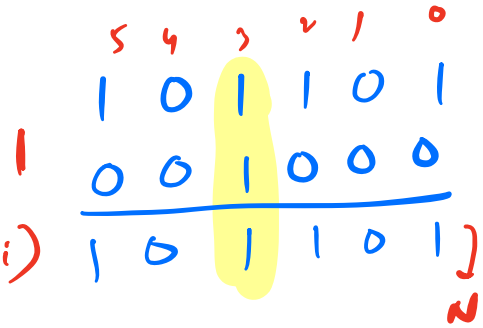
if((N & (1<<i)) > 0)

↳ ith bit is SET

else

↳ ith bit is UNSET



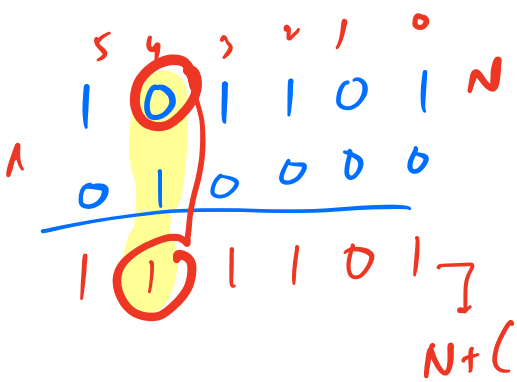


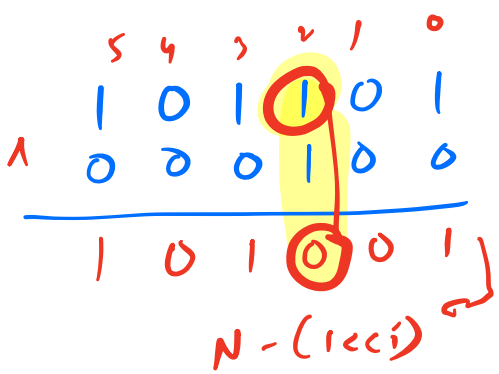
if((N | (1<<i)) > N)

↳ ith bit is UNSET

else

↳ ith bit is SET

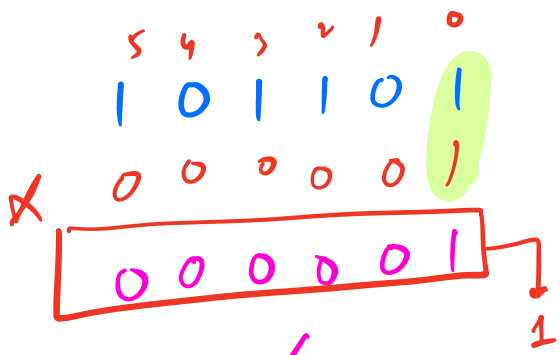


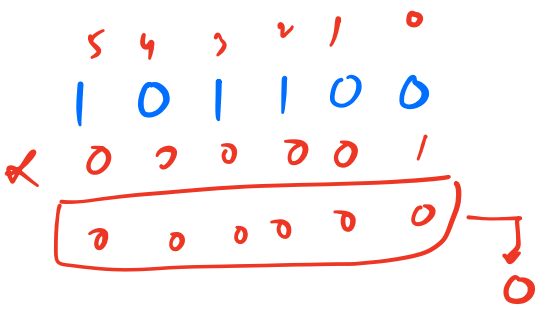


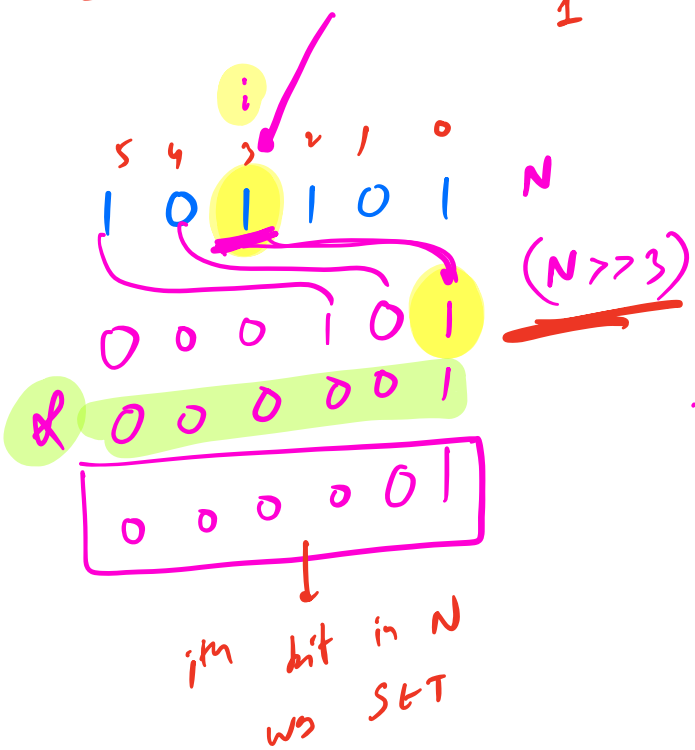
if((N & (1<<i)) < N)

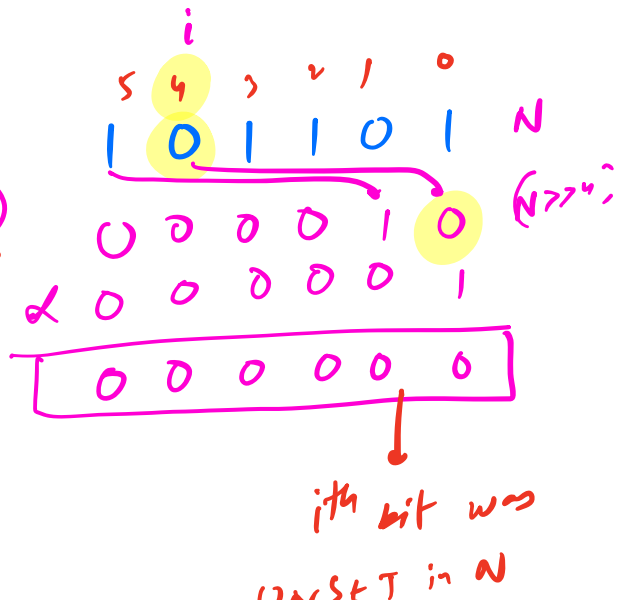
↳ ith bit is SET

else ⟶ NOT SET









if $(((N >> i) \& 1) > 0)$

↳ ith bit in N was SET

else

↳ UNSET

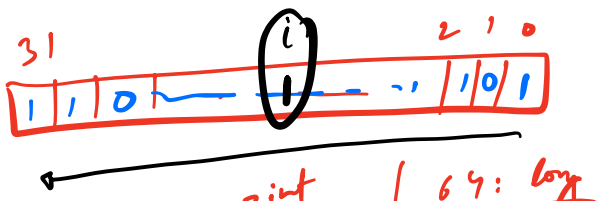
Q Given a number N. Count the total no. of set bits!

$N = (6)_{10} \longrightarrow (110)_2$: 2

$N = (7)_{10} \longrightarrow (111)_2$: 3

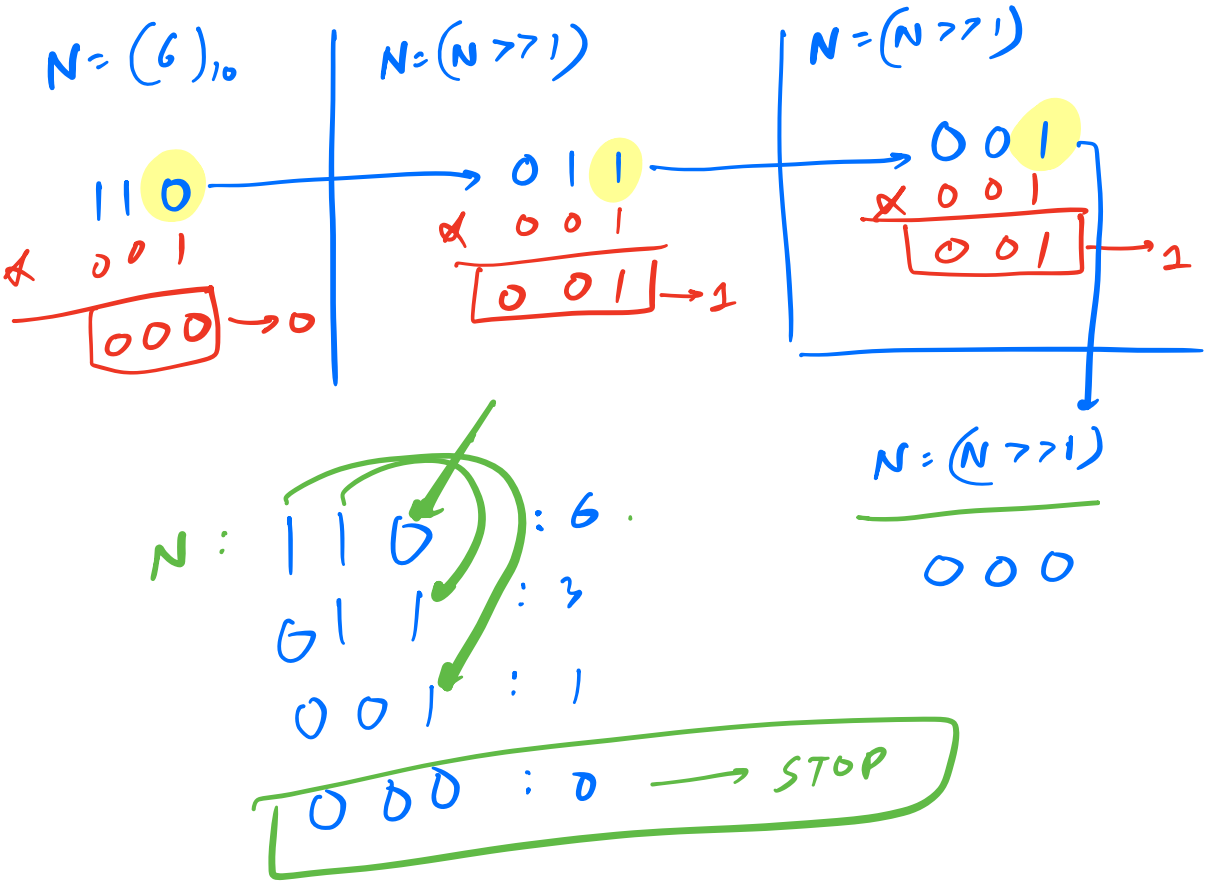
$N = (12)_{10} \longrightarrow (1100)_2$: 2

N: int → 32 b



int | 64: long

```
cnt = 0
for(i=0; i<32; i++){
    if((N & (1<<i)) > 0){
        cnt++;
    }
}
ret cnt;
```



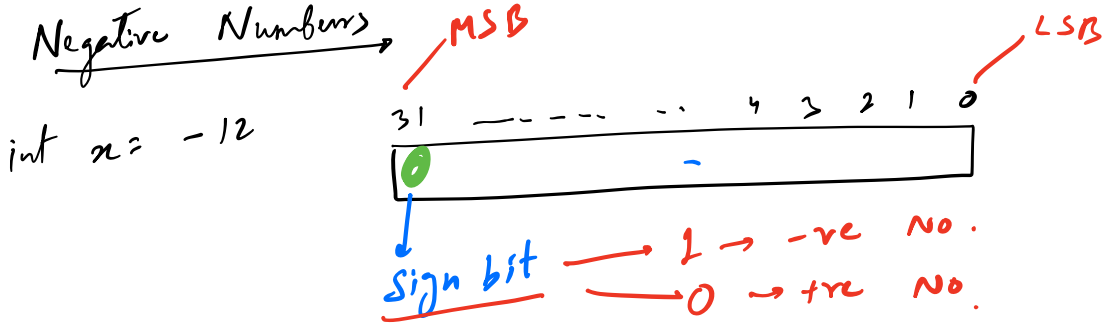

```
cnt = 0;
while (N > 0) {
   if ((N & 1) > 0){
        cnt++;
   }
    N = (N >> 1);
}
```

TC : O(log N)

SC : O(1)

Negative Numbers

int x = -12



8 bit No

byte x = -12

	7	6	5	4	3	2	1	0
12 :	0	0	0	0	1	1	0	0
Toggle all bits :	1	1	1	1	0	0	1	1 → 1's complement
Add 1 :	+							
	0	0	0	0	0	0	0	1
-12 →	1	1	1	1	0	1	0	0 → 2's comp

$-2^7 + 2^6 + 2^5 + 2^4 + 2^2$

$-128 + 64 + 32 + 16 + 4 = -12$ //

0 0 0 0 1 1 0 0

$0 \times -2^7 + 0 \times 2^6 + 0 \times 2^5 \ldots + 1 \times 2^3 + 1 \times 2^2 + \ldots$

N = 0 : 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1

\+ 0 0 0 0 0 0 0 1

-0 → 0 0 0 0 0 0 0 0

N: 45 0 0 1 0 1 1 0 1

1 1 0 1 0 0 1 0

\+ 0 0 0 0 0 0 0 1

\+ 1 1 0 1 0 0 1 1 → -45

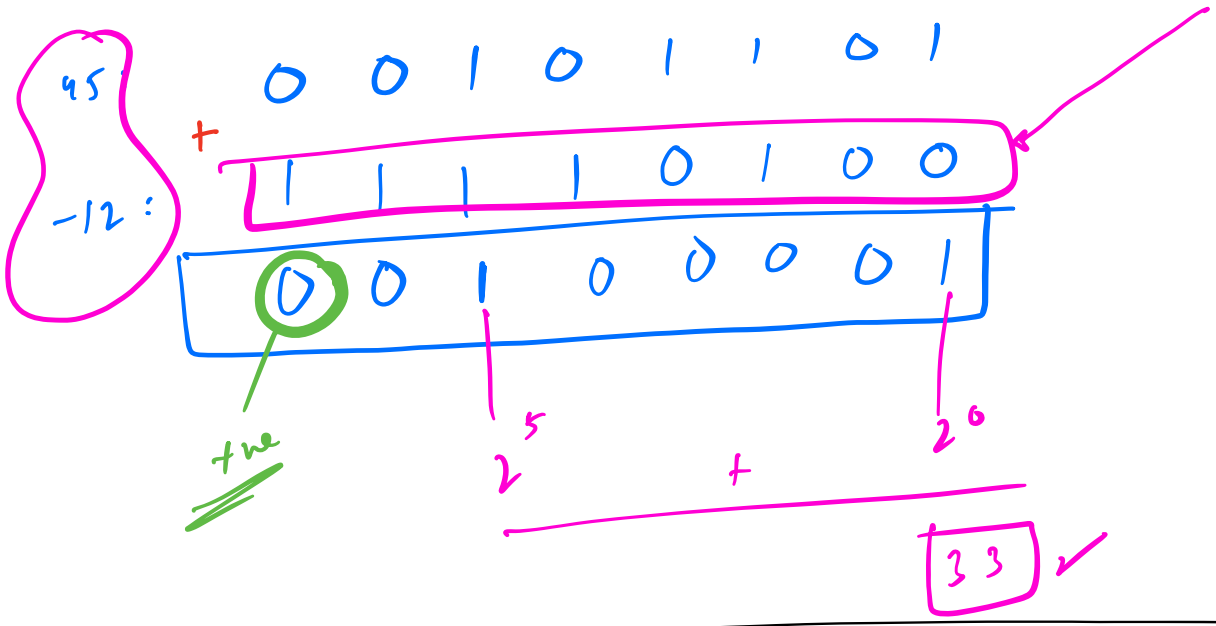
\+ 0 0 1 0 1 1 0 1 → 45

0 0 0 0 0 0 0 0 → 0

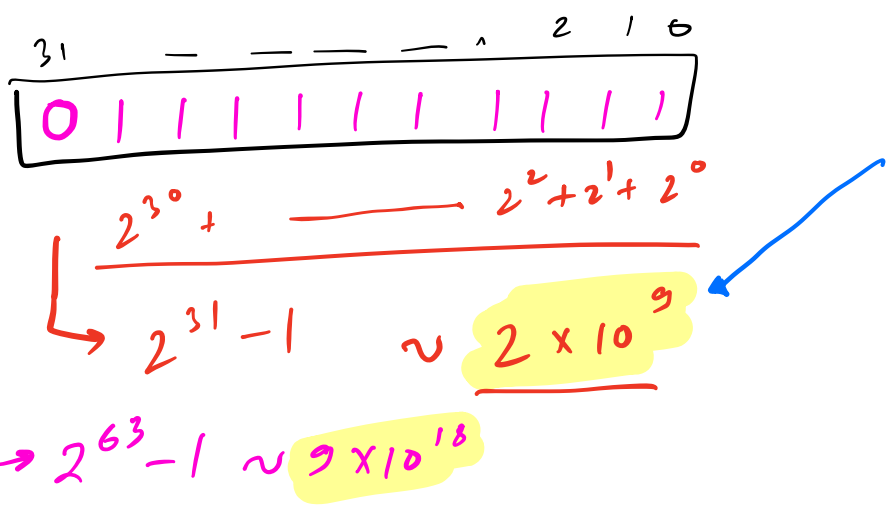
1 Subtraction of binary Numbers →

45 − 12

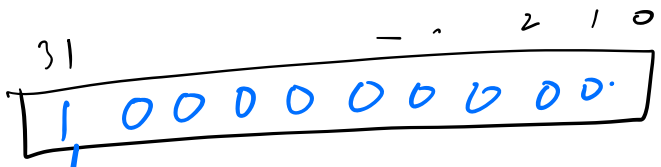
45 + (−12)



⑧ MAX no. an int can store ?



(a) MIN no. an int can store?



$(-2^{31}) \sim -2 \times 10^9$

long → $-2^{63} \sim -9 \times 10^{18}$

(b) MAX no an uint can store?

$2^{31} + 2^{30} + 2^{29} + \cdots + 2^0$

$\hookrightarrow 2^{32} - 1$

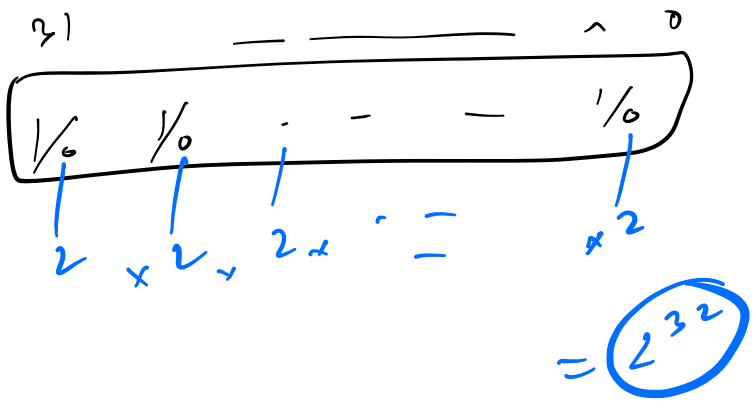
$\sim 4 \times 10^9$

(c) MIN no an uint can store?

$\hookrightarrow 0$

uint $[0, 2^{32}-1]$ — 2^{32}

int $[-2^{31}, 2^{31}-1]$ → 2^{32}



$= 2^{32}$

Q Given an array. Find the sum of the Array!

10^9

```
int sum = 0;
for (i=0; i<N; i++) {
    sum += A[i];
}
ret sum;
```

long 10^{18}

CONSTRAINTS →

$1 <= N <= 10^5$

$1 <= A[i] <= 10^9$

10^9 10^9 . . . 10^9

$10^9 \times 10^5$

10^{14}

1 1 1
0 1 1 1 → 7
+ 0 0 0 1
1 0 0 0 → -8

overflow

1 Given 2 ints a & b, get a * b!

int ans = a x b;
get ans;

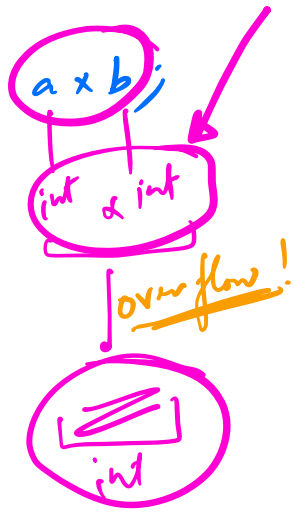
~10^9 x ~10^9

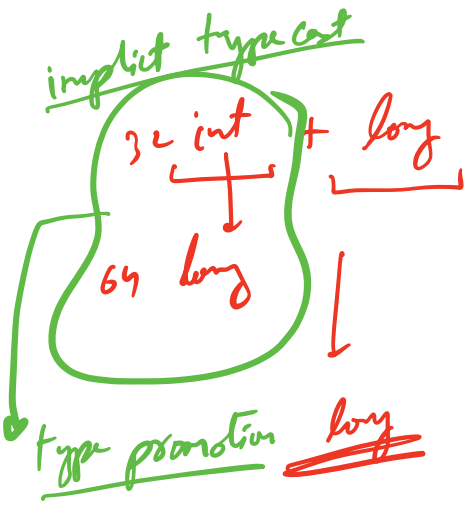
~10^18

✗

long ans = a x b;
get ans;

✗





long ans = long (a x b)

overflow!

Explicit typecast

✓ long ans = long(a) x b

or

✓ ——— = long(a) x long(b)

✓ ——— = a x 1L x b