Decimal Number System

→ Also known as base-10 System, the system we use in everyday lives.

→ Called base-10 because a single digit can take values from 0 to 9 (10 digits)

→ Position of each digit represents a power of 10.

Ex:-

$342 = 300 + 40 + 2$

$= 3 * 10^2 + 4 * 10^1 + 2 * 10^0$

$2563 = 2000 + 500 + 60 + 3$

$= 2 * 10^3 + 5 * 10^2 + 6 * 10^1 + 3 * 10^0$

Binary Number System

→ Also known as base - 2 system, is used in digital electronics & computing.

→ Only two digits, 0 & 1

→ Each digit represents a different power of 2.

EX:-

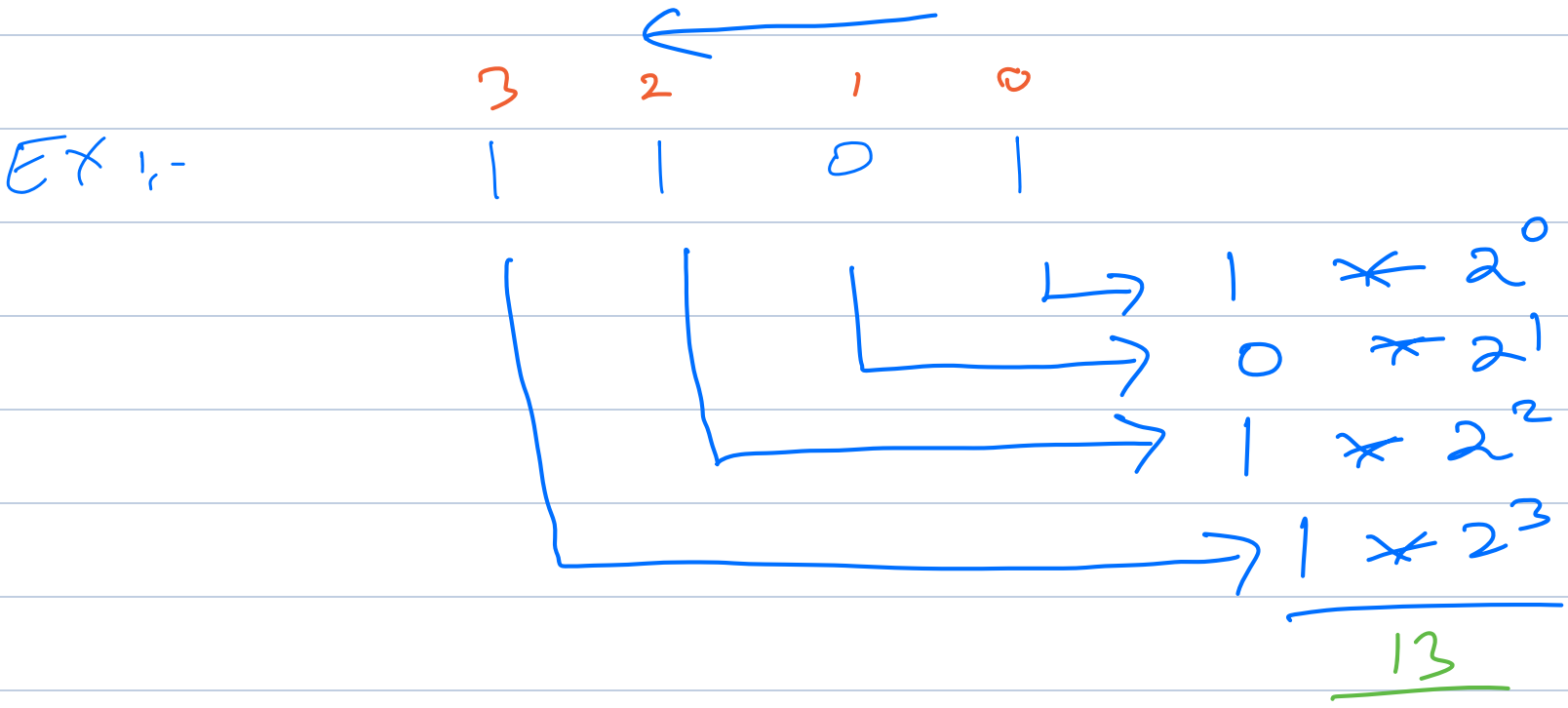
$(110)_2 \rightarrow 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

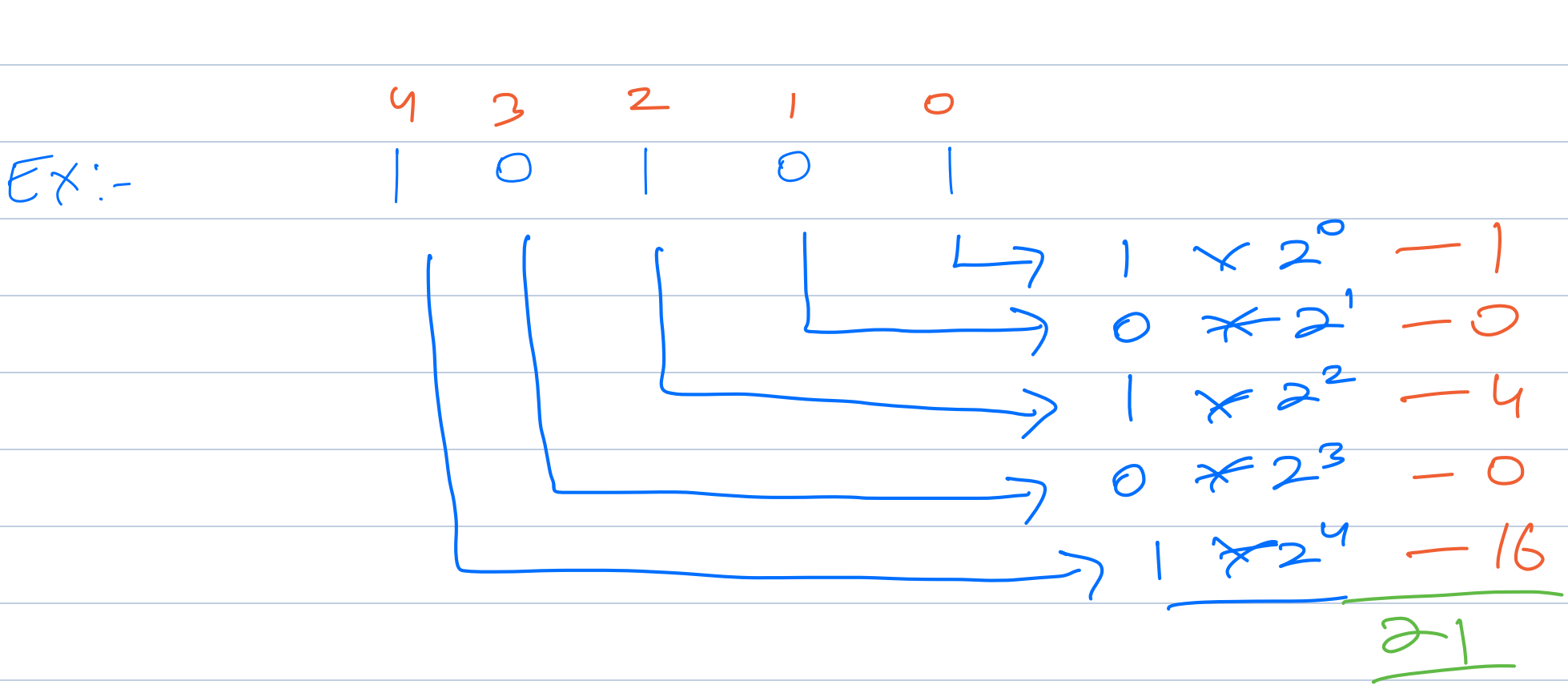
$\rightarrow 4 + 2 = 6$

$(1011)_2 \rightarrow 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

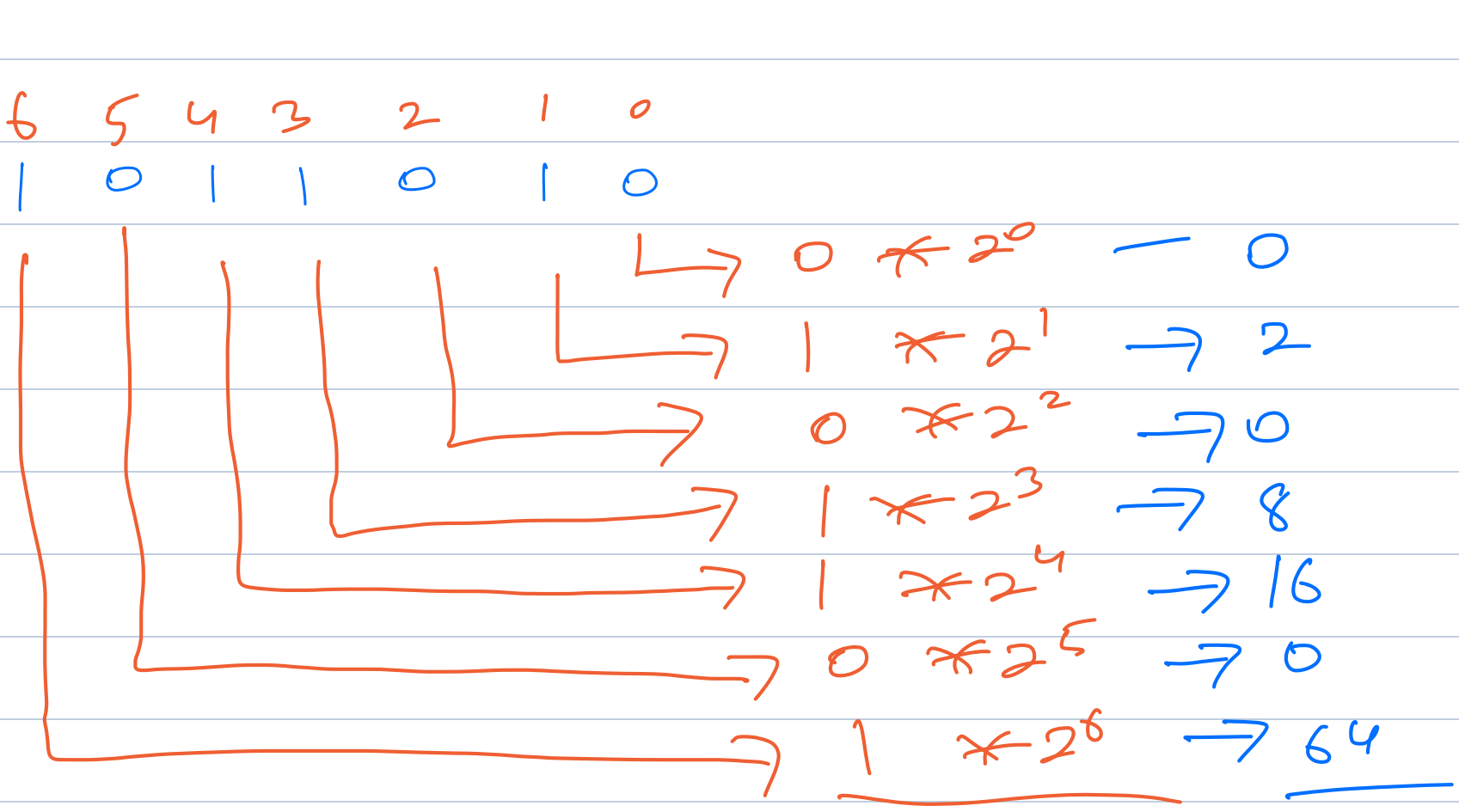
$\rightarrow 11.$

Binary To Decimal Conversion





Quiz 1



Decimal To Binary Conversion

→ Using long division method

Ex :- $(20)_{10} = (?)_2$

2	20	0
2	10	0
2	5	1
2	2	0
2	1	1
	0	

→ (10100)

Quiz 2

$(45)_{10} = (?)_2$

2	45	1
2	22	0
2	11	1
2	5	1
2	2	0
2	1	1
	0	

101101

Addition of Decimal Numbers

```
   +1  +1
    3   6   8
+   2   5   3
-------------
    6   2   1
```

digit	carry forward
11 / 10	11 / 10
12 / 10	12 / 10
6 / 10	6 / 10

Addition of Binary Numbers

Ex:-

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

digit	carry forward
2 / 2	2 / 2
1 / 2	1 / 2
2 / 2	2 / 2
2 / 2	2 / 2
2 / 2	2 / 2
1 / 2	1 / 2

Quiz 3

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

digit	carry forward
1 / 2	1 / 2
2 / 2	2 / 2
3 / 2	3 / 2
1 / 2	1 / 2
1 / 2	1 / 2

Bitwise Operators

→ Operators used to perform operations on individual bits of binary numbers.

→ Used to manipulate binary data.

→ 0 → false / unset
1 → true / set

AND (&)

→ Performs logical AND operation on each pair of corresponding bits.

→ Returns 1 only if both the bits are set (1) else return 0.

			Result
0	&	0	0
0	&	1	0
1	&	0	0
1	&	1	1

OR (|)

→ Performs logical OR operation on each pair of corresponding bits.

→ Returns 1 if either of the bits are set (1) else return 0.

		Result
0	0	0
0	1	1
1	0	1
1	1	1

XOR (^) (exclusive OR)

→ Performs logical XOR ^ operation on each pair of corresponding bits.

→ Returns 1 if corresponding bits in Input are different else return 0.

		Result
0	0	0
0	1	1
1	0	1
1	1	0

NOT (! / ~)

→ Takes a single binary number as input & performs logical NOT on each bit

→ Resulting bit is opposite to the input bit.

	Result
~ 0	1
~ 1	0

1 ↑ 0 ↑ 1

1 — 0

Examples

① 5 & 6

= 4

```
  1 0 1
& 1 1 0
-------
  1 0 0   - 4
-------
```

② 20 & 45

= 4

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

③ $92 \wedge 154 = 198$.

$$\begin{array}{r} 01011100 \\ \wedge\ 10011010 \\ \hline 10001110 \end{array}$$

④ ~ 01011100

$$\begin{array}{r} 10100011 \end{array}$$

$$\begin{array}{r} 00000111 \\ \sim\ 11111000 \end{array}$$

⑤ $92 \mid 154$

$$\begin{array}{r} 01011100 \\ \mid\ 10011010 \\ \hline 11011110 \end{array} = 222$$

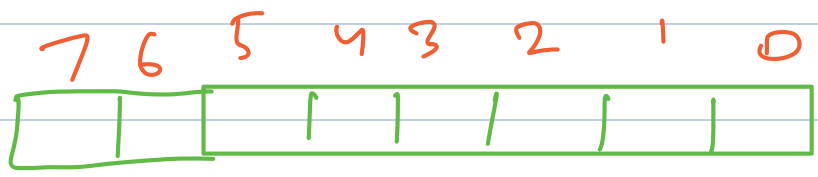
Quiz 4

$20 \wedge 45$

$$\begin{array}{r} 010100 \\ \wedge\ 101101 \\ \hline 111001 \end{array}$$

57

Binary Representation of Negative Numbers



-5

Two's Complement of a Number:

(1) Convert absolute value of the number to its binary representation.

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

(2) Invert every bit in the result of step 1.

1	1	1	1	1	0	1	0

(3) Add 1 to the final result.

1 1 1 1 1 0 1 0

+ 1

MSB ↘ 1 1 1 1 1 0 1 1 (−5)

Signed bit

→ 1×2^0

→ 1×2^1

→ 0×2^2

→ 1×2^3

→ 1×2^4

→ 1×2^5

→ 1×2^6

→ $(-)(1 \times 2^7)$

$1 + 2 + 8 + 16 + 32 + 64 - 128$

$= -5$

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

(a)

-2^7

$2^7 - 1$

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

(b)

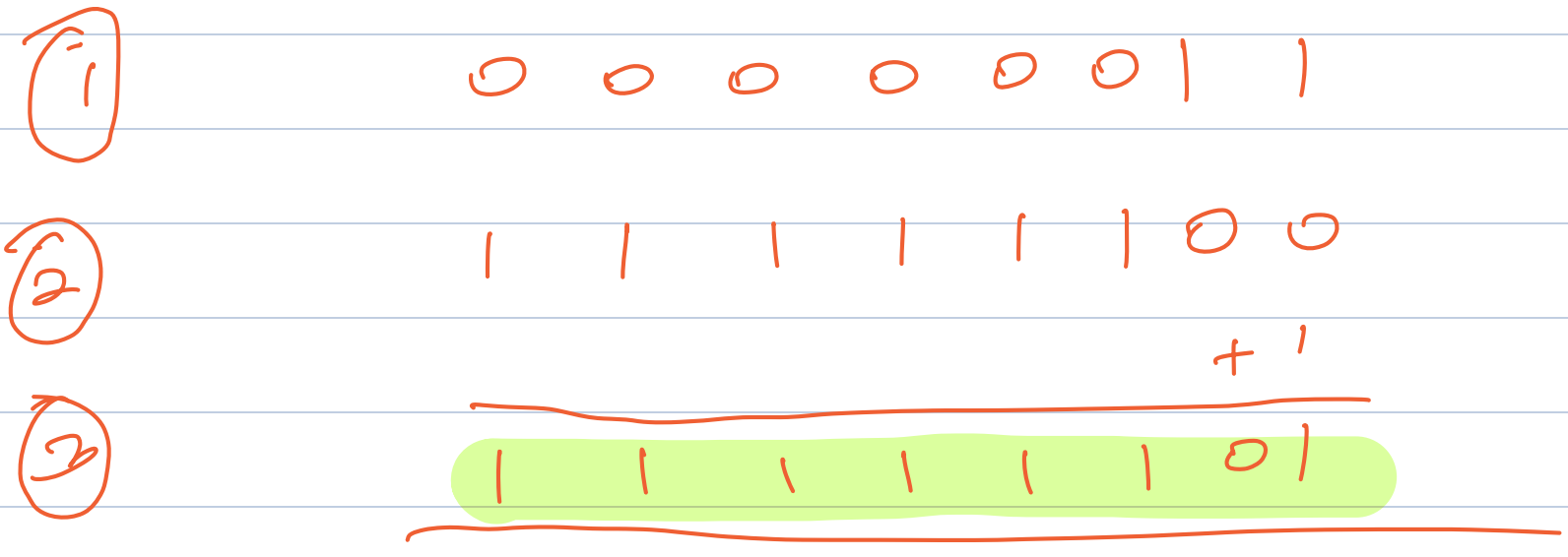
$$a\left(\frac{r^n - 1}{r - 1}\right) \qquad = 1\left(\frac{2^7 - 1}{2 - 1}\right)$$

$= 2^7 - 1$

If the MSB is 1, it is a negative number else positive.

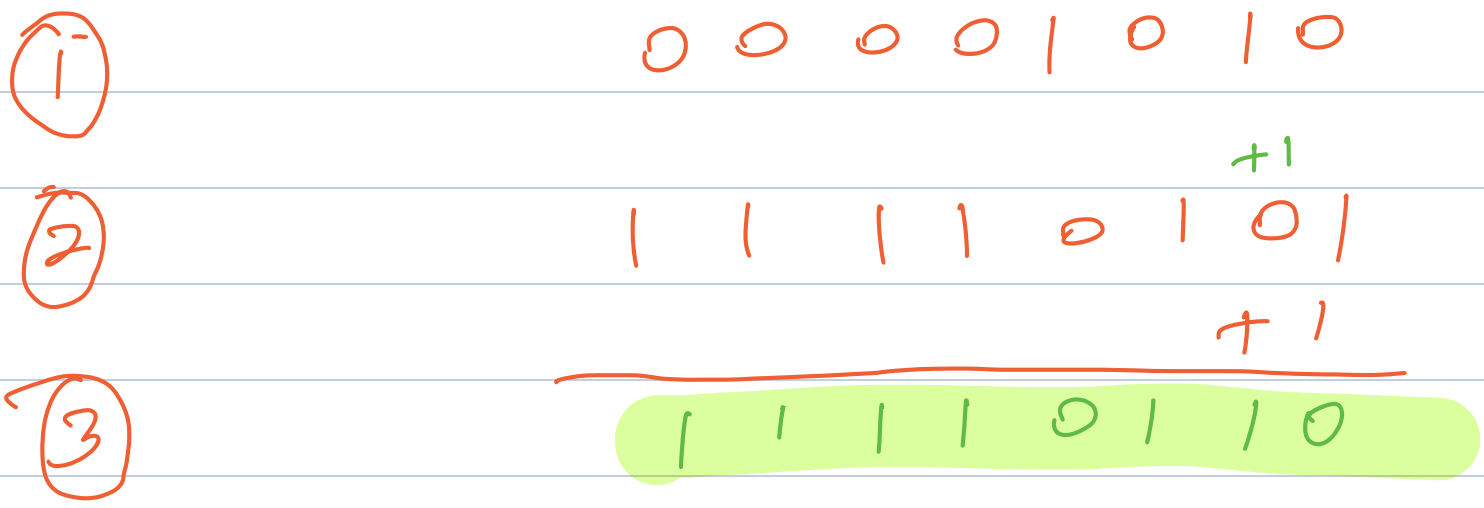
Quiz 5

−3 in 8-bit signed integer format.



Quiz 6

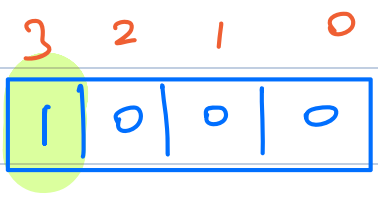
−10 in 8-bit signed integer format.



8:30 :-

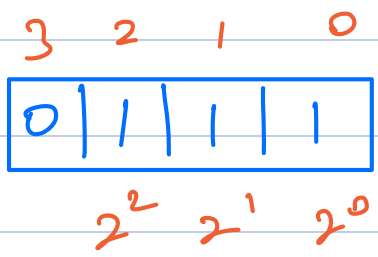
Range of data bytes

Smallest



$-2^3 = -8$

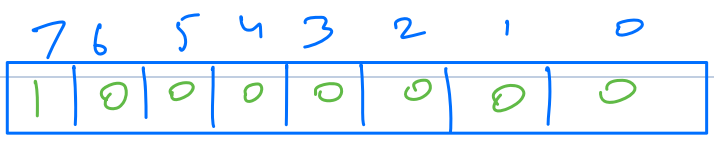
Largest.



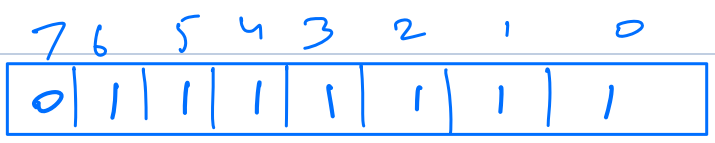
$2^3 - 1 = 7$

$$-2^3 \leq x \leq 2^3 - 1$$

Smallest:



$-2^7 = -128$



$2^7 - 1 = 127$

If I have a datatype of n bits.

$$-2^{N-1} \leq x \leq 2^{N-1} - 1$$

Int → 4 Bytes. → 32 bits.

$$-2^{31} \leq \text{Int} \leq 2^{31} - 1$$

Approximately:

$$-2 * 10^9 \leq \text{int} \leq 2 * 10^9$$

long → 8 byte → 64 bits

$$-2^{63} \leq \text{Any long} \leq 2^{63}-1$$

$$-9 * 10^{18} \leq \text{long} \leq 9 * 10^{18}$$

Importance of Constraints

$a = 10^5$

$b = 10^6$

Trying to compute the value of $a * b$.

① `int c = a * b;` ✗

② `long c = a * b;` ✗ (Because of ALU)

③ `long c = long (a * b)` ✗ (Because of ALU)

④ `long c = long (a) * b;` ✓

Q:- $1 \le N \le 10^5$

$1 \le A[i] \le 10^6$

Given an array of size N, calculate the sum of array elements!

long

~~int~~ sum = 0;

for (int i = 0; i < N; i++) {

sum += A[i];

} // sum = sum + A[i];

return sum;

✓

Cotril → 21st march.

Friday.

Next week Monday

Tuesday - Cotril

Next week:

Wednesday. 9 of.

Doubts

Commutative.

$$a + (b + c) = (c + a) + b$$

Associative.

$$a + (b + c) = (a + b) + c$$

$\downarrow$ | $\curvearrowright$ $\sim$

1 1 0 9

-3

8

$+ \ (-3)$

1 1 1 1