Q. Given 2 numbers, a and b, calculate the LCM of the 2 numbers. [LCM → least common multiple).

Ex → [a = 9, b = 12

We already know how to calc the gcd / hcf.

ans → LCM → 36

G → gcd of a, b L → LCM of a, b

$a \times b = G \times L$

$$\text{lcm} = \frac{a \times b}{\text{Or}} \leftarrow$$

gcd(a,b)

$$\text{lcm}(a,b) = \frac{a \times b}{\text{gcd}(a,b)}$$

Q. Given a number x, check if the no. is a palindrome or not.??

NAMAN

Eg → 1441 → Yes

141 → Yes

161 → Yes

9 → Yes

153 → NO

44 → Yes

"hello" "World"

What is a String? → It is a collection of chars.

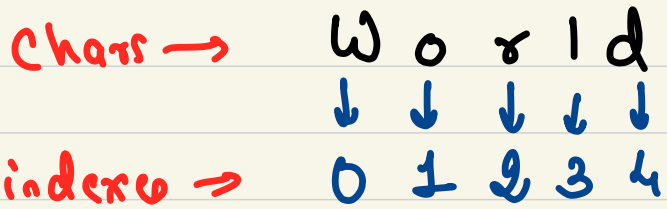
↳ In JS we can individually access chars of a string

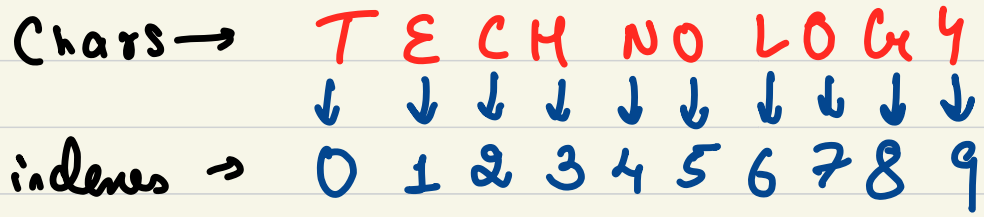
Chars → h e l l o

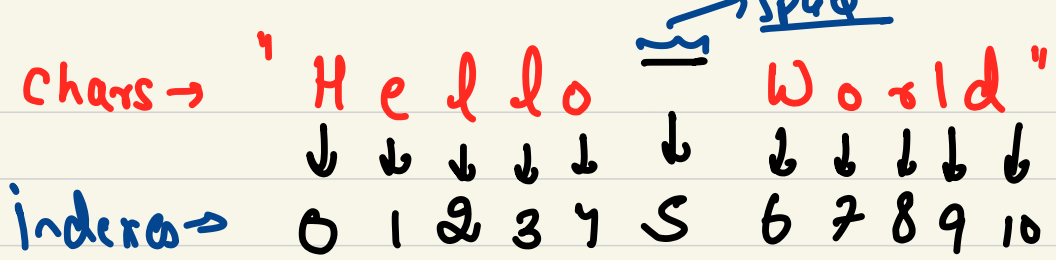
indexes → 0 1 2 3 4 → indexes act as a position

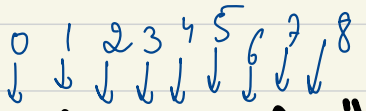
↓

It is a unique mapping for all the chars of string to a unique no. the indexes in JS starts with 0.


Chars →
W o r l d
indexes →
0 1 2 3 4


Chars →
T E C H N O L O G Y
indexes →
0 1 2 3 4 5 6 7 8 9





let ch = str[6];

↳ index

"o"

let m = str[2];

↳ "c"

if we know the index we can extract out the char

str[i]

- str → name of the string variable
- i → number

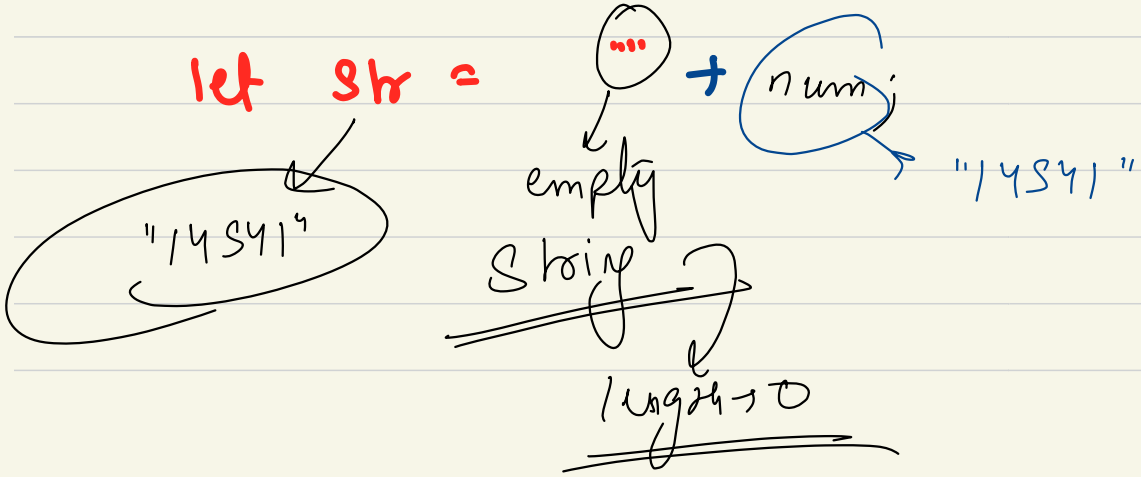
→ 1 4 5 4 1 → No

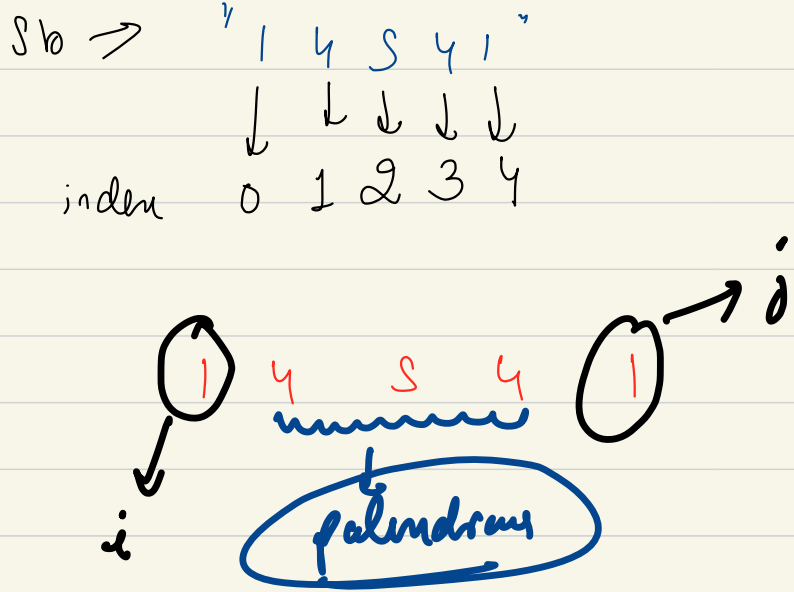
Numbers cannot be traversed using indexes.

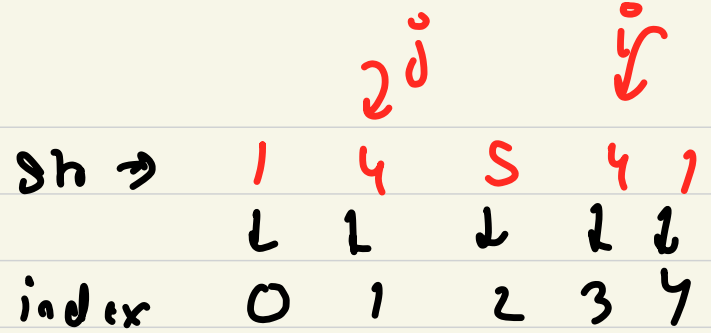
→ 14541 ⟶ "14541"

let num = 14541;

let str = "" + num;




Sb →
" 1 4 S 4 1 "
index 0 1 2 3 4
1 4 S 4 1
i
i
palindrom

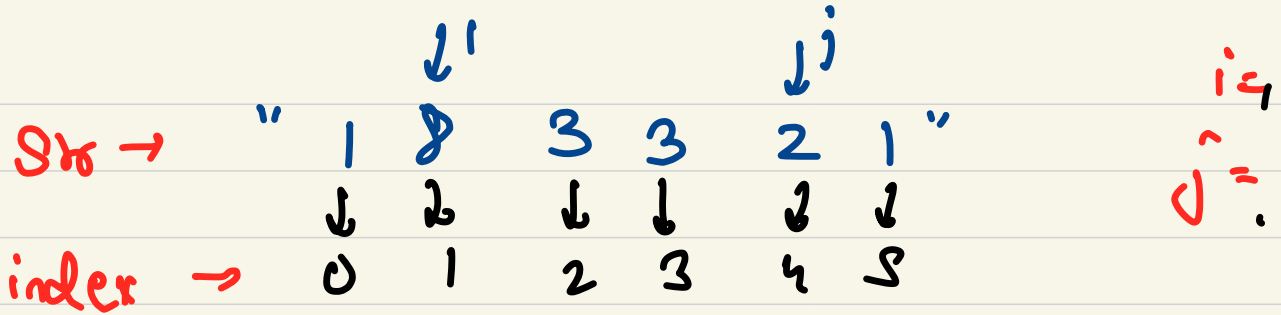


i = 0 // 2

j = str.length-1 // 2

this comparison needs to be done again & again

```
if (str[i] == str[j]) {
    i++
    j--
}
```




```
while (i<=j) {
    if (str[i] == str[j]) {
        i++;
        j--;
    } else {
        return false;
    }
}
return true;
```

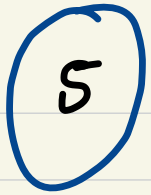
Q. Given a decimal number x, convert it into a binary number. (we can return it in a form of a string)

↳ x = 5

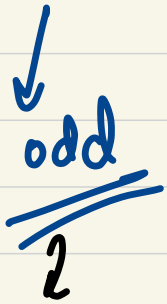
ans → "101"

x = 15

ans → "1111"



→ 101



0 → 000
1 → 001
2 → 010
3 → 011
4 → 100
5 → 101
6 → 110

because 5 is odd

it doesn't matter what are

the remaining bits, but the last bit

will be 1.

6.5 → 6

15 → 1111

$\lfloor \frac{15}{2} \rfloor$

2

7 → 111

6 → 110

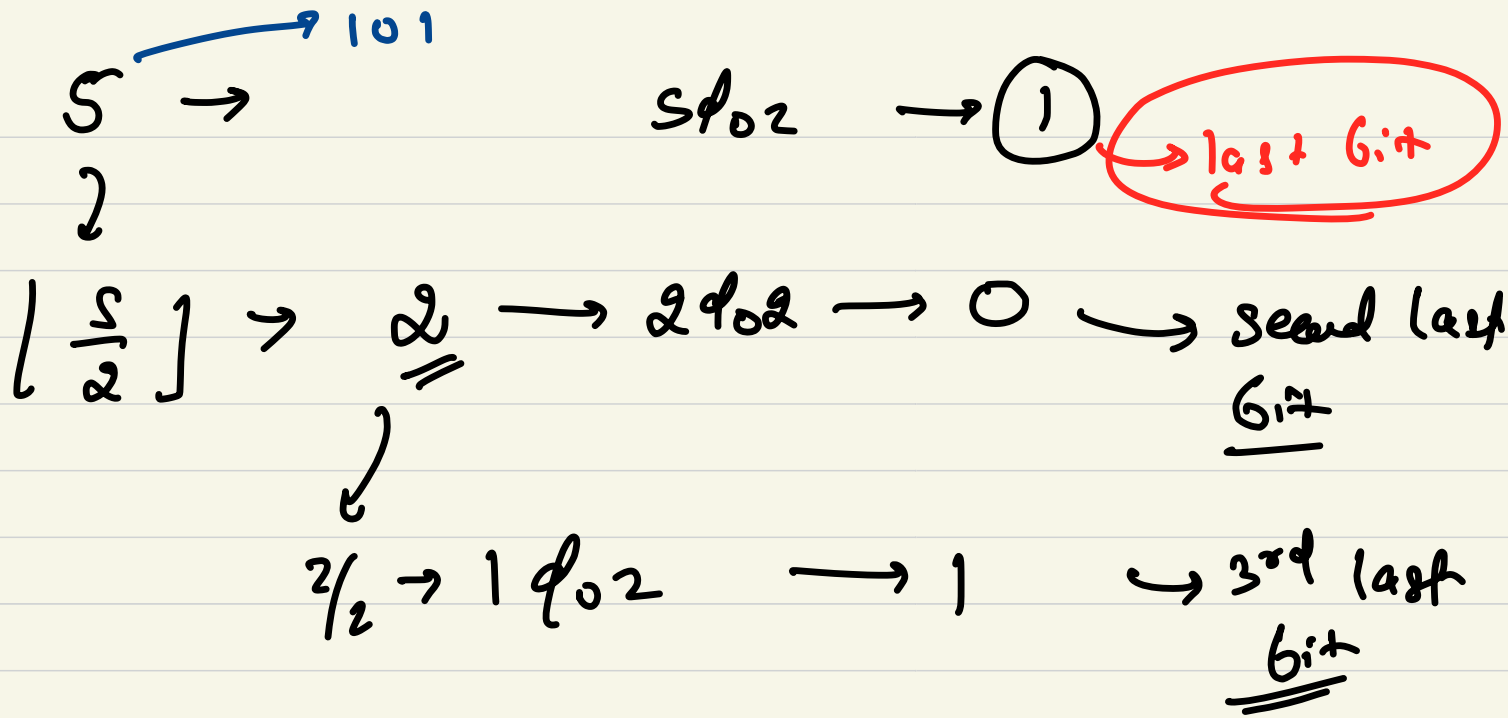
2

3 → 11

5 → 101

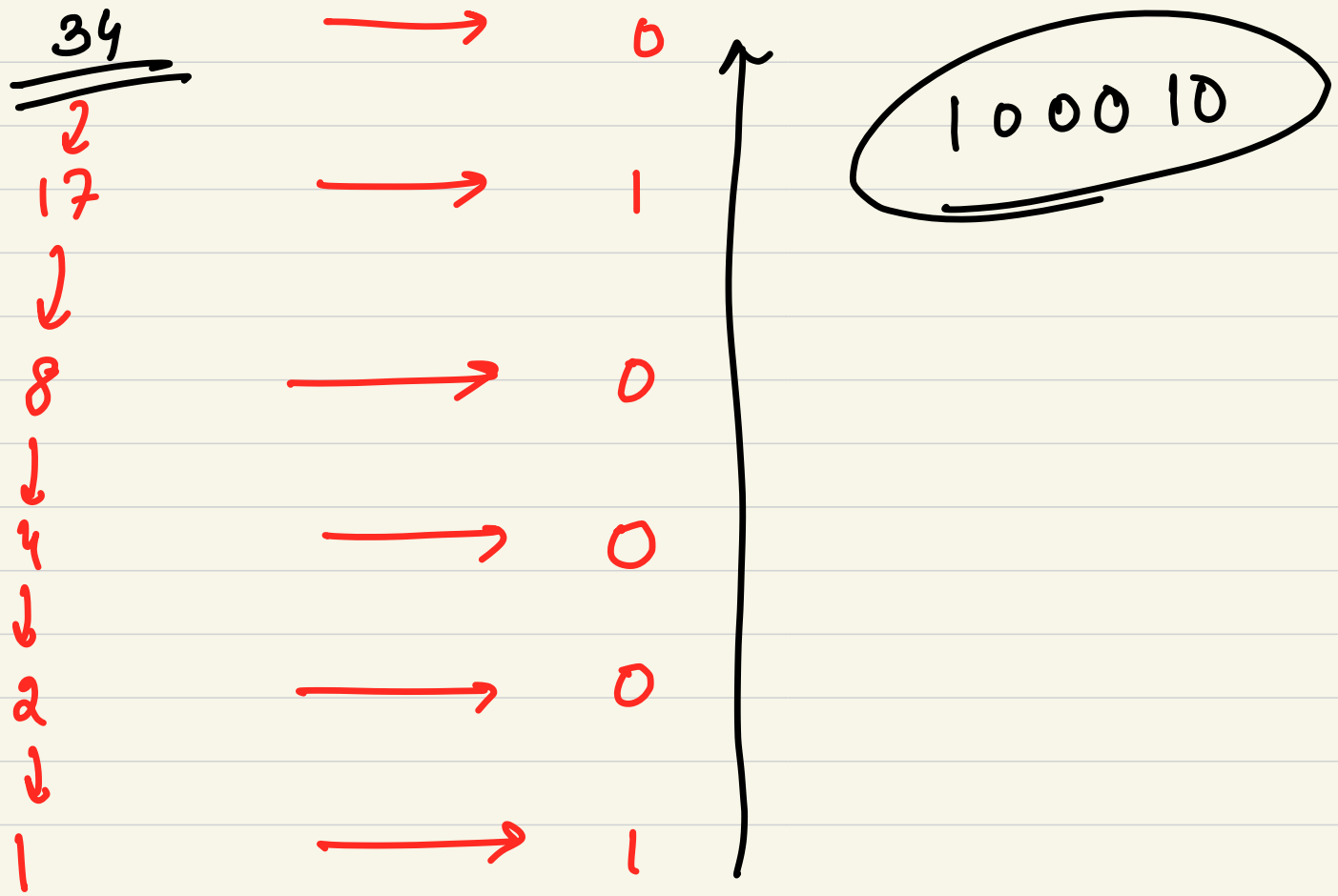
1

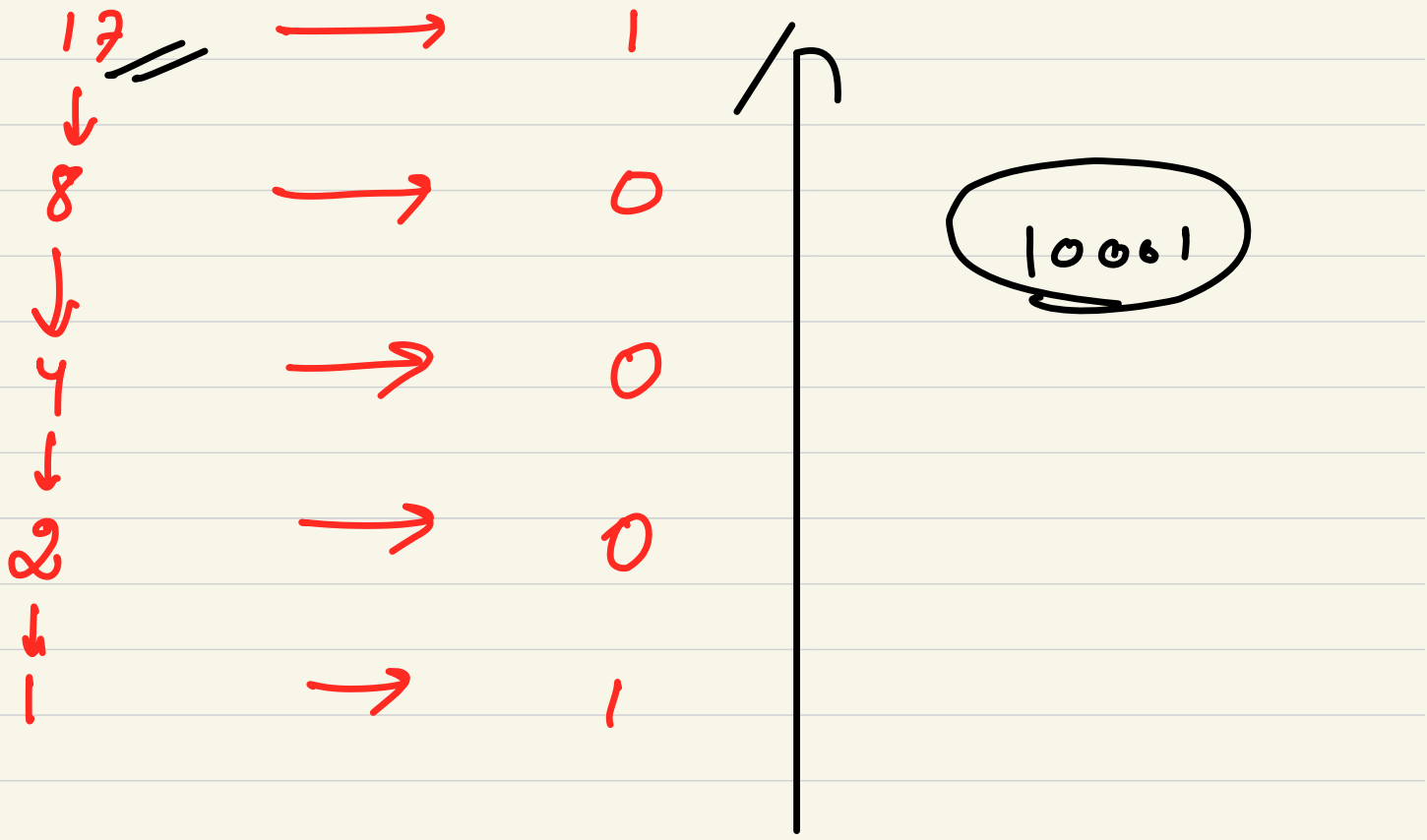
$\lfloor \frac{5}{2} \rfloor \rightarrow 2 \rightarrow 10$

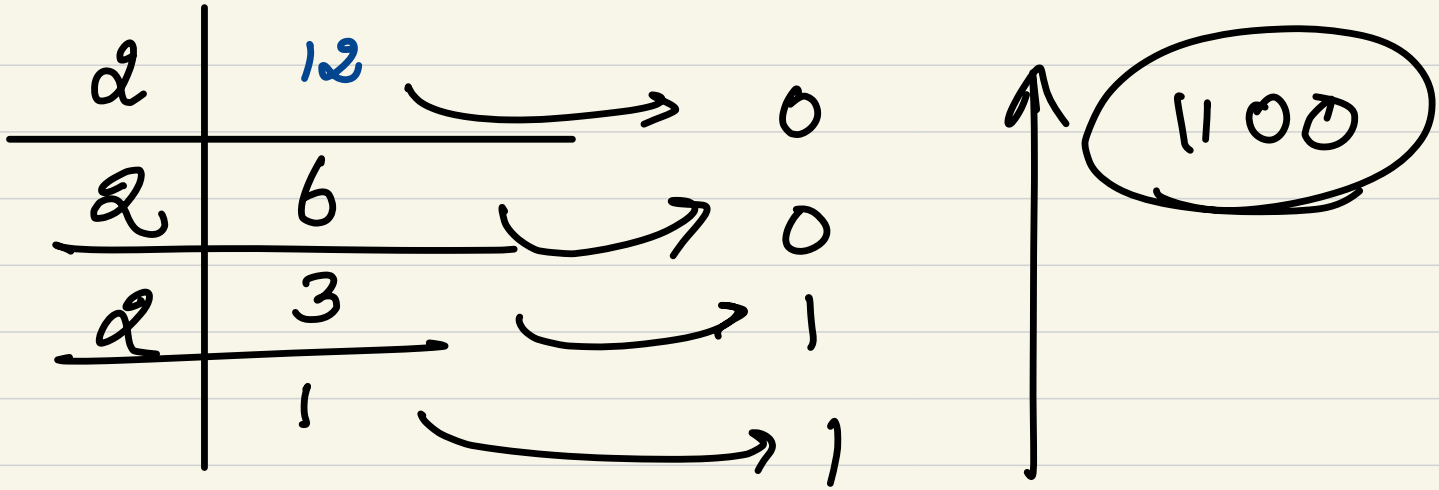

101
5 → 5%2 → 1
last bit
⌊5/2⌋ → 2 → 2%2 → 0 → second last bit
2/2 → 1%2 → 1 → 3rd last bit

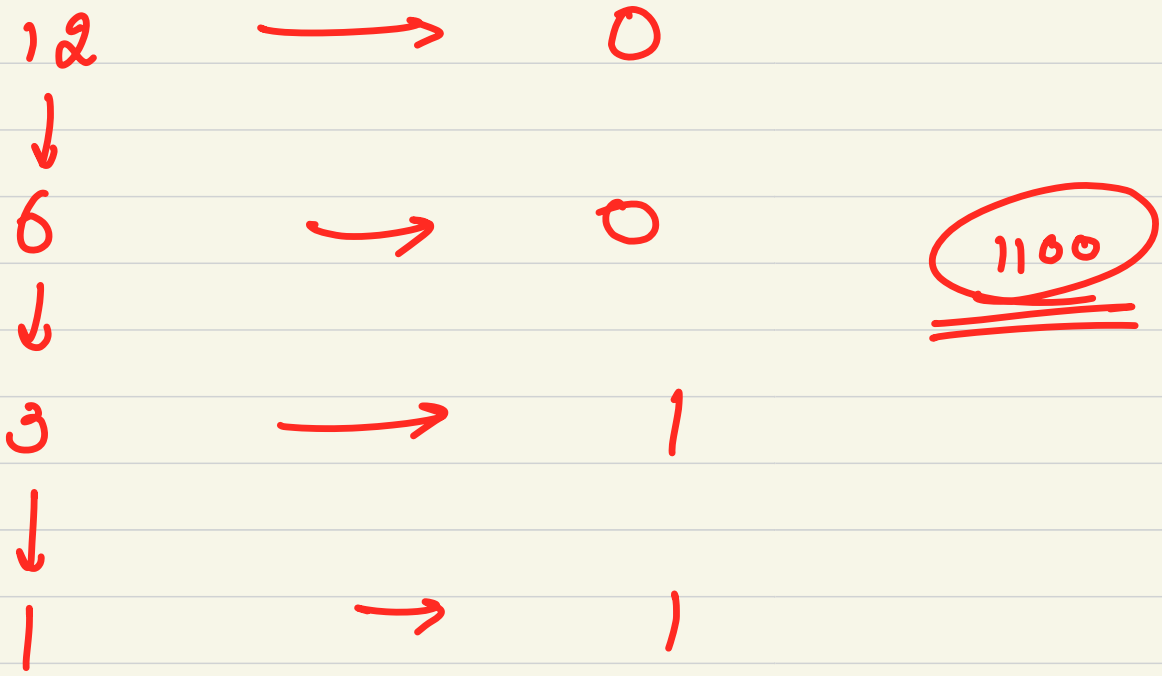
whatever no. we have, we can check if the no. is odd or even. Based on this we can delect the last.

Now the remaing bits can be calculated using no./2 value.


34
2
17
2
8
2
4
2
2
2
1
0
1
0
0
0
1
1 0 0 0 10


17
8
4
2
1
1
0
0
0
1
10001


2
12
0
2
6
0
2
3
1
1
1
1100


12
0
6
0
3
1
1
1
1100