Bit Manipulation

Binary Numbers.

$(\underset{2}{1}\underset{1}{0}\underset{0}{1})_2 \rightarrow 1\times2^2 + 0\times2^1 + 1\times2^0$

$= 4 + 0 + 1$

$= 5.$

* computess works on binary numbers.

for decimal to binary.

ex. 14 take LCM =

2	14	0
2	7	1
2	3	1
	1	

$= (1110)_2.$

ex. 13.

check whether it is enen or odd.

odd ∴ $(1101)_2$ $\frac{\cancel{13}}{\cancel{2}}$ $\frac{\cancel{6}}{2}$ $\frac{\cancel{3}}{2}$ $\frac{\cancel{1.5}}{\cancel{2}}$ 0.5

* To find negatine inverse in binary :→

2's complement (negative inerse).

ex. 5. (101)

step-1 : Invert all bits. = 010

step-2 : Add 1 :

$$\begin{array}{r} 010 \\ +\;\;1 \\ \hline 011 \end{array} \rightarrow \text{2's complemen of 101 i.e. } (-5).$$

* Subtraction. of binary numbers.

ex : 12-5 → negative inverse

i.e. 12+(-5) i.e. 2's complement.

$$= \begin{array}{r} 1100 \\ +\textcircled{1}011 \\ \hline 0111 \end{array}$$

?

☆ **Bitwise Operators :→**
(&, |, ^, ~, >>, <<)

when both are different
↑ then (1)

1) AND operator (&), OR (|), XOR (^) operators.

a	b	a&b	a|b	a^b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

2) Right shift operator (>>) :→.

ex. 12>>2

step-1 : convert 12 to binary = 1100

↳ 1 1 0 0 (shift to right by 2)

0 0 1 1

↳ 3

8] left shift operator (<<)

12<<2

↳ 1 1 0 0

↳ 110000

☆ programming approach of right shift and left shift operator.

```
ex.   int  a = 5;
      int  b = a>>1;
```

// whenever we are using right shift operator
// then we are dividing that no. by 2.

ex. 5 → 101
2 → 010
1 → 001
0 → 000

// application of (<<) : 1] Whenever we are using loop & in it wee are dividing it by 2, then instead of division we can use right shift operator.

4] left shift operator (>>):

```
int a = 3;
int b = a<<1;   (left shift means multiplying
                  the number by 2)

3 → 1 1
6 → 1 1 0
12 → 1 1 0 0
```

ex. program to find no is even or odd.

~~if (a>>2~~.

by using bit masking concept.

```
2 → 1 0      } even
4 → 1 0 0    } ~~odd~~ no
6 → 1 1 0    } last bit always 0.
-----------------
3 → 1 1      } odd no.
7 → 1 1 1    } last bit
9 → 1 0 0 1  } always 1.
```

if we do AND operation of given no. with 1.

ex. 7 (1 1 1)

```
   1 1 1
 & 0 0 1
 -------
   0 0 1 → last bit = 1, then it is ~~even~~ odd no.
```

ex. 6 (1 1 0)

```
   1 1 0
 & 0 0 1
 -------
   0 0 0 → last bit = 0, then it is even no.
```

```
∴ if (a&1==0)
  {  No. is even;
  }
  else
  {  odd;
  }
```

★ Bit masking :→

- find ith bit :

8 7 6 5 4 3 2 1 0

ex n = 1 0 0 1 1 0 1 0 1

find ~~th~~ bit, therefore
i=5th

find which bit is present at i=5.

for finding right shift the bit present at i=5.

mask = 0 0 0 1 0 0 0 0 0

perform & operation to find whether the bit is non-zero or one.

n = 1 0 0 1 1 0 1 0 1

& mask = 0 0 0 1 0 0 0 0 0

0 0 0 1 0 0 0 0 0

if it ~~is~~ all zero then, the ~~o~~ bit we want to find is 0.

or if it ~~is~~ is all one then, the bit we want to find is 1.

- set ith bit :

8 7 6 5 4 3 2 1 0

n = 1 0 0 1 1 0 1 0 1

or

mask = 0 0 0 0 0 1 0 0 0

(left shift 1 three times)

1 0 0 1 1 1 1 0 1

mask = 1<<i

n = n | mask

- clear ith bit

8 7 6 5 4 3 2 1 0

n = 1 0 0 1 1 0 1 0 1

1<<i = 0 0 0 0 1 0 0 0 0

mask. ~(1<<i) = 1 1 1 1 0 1 1 1 1

n = 1 0 0 1 1 0 1 0 1

& mask = 1 1 1 1 0 1 1 1 1

1 0 0 1 0 0 1 0 1

ith bit changed.

- find number of bits to change to convert a to b.

Question : you have given two binary numbers.

ex. a → 10110

b → 11011 , so for converting a into b, how many bits you have to chang.

XOR

0	0	0
0	1	1
1	0	1
1	1	0

ex. ① 5^5 = 0.

② 0^n = n

} XOR property.

Q1] Find the only non-repeating element in an array where every element repeats twice.

ex. a = {5,4,1,4,3,5,1}

statement : We have to return only non-repeating element. i.e. 3 in given array.

Three ways to solve this problem.

1] → using nested loop. [time complexity = O(n²)]

2] → we use space here, hash map. [space complexity = O(n²)]

3] → using XOR property. [time complexity O(n)
space complexity O(1)]

a = {5,4,1,4,3,5,1}

int res = 0;

in loop { res^a[0] = 0^5 = 5

res^a[1] = 5^4

res^a[2] = 5^4^1

res^a[3] = $5 \wedge \cancel{4} \wedge 1 \wedge \cancel{4} = 5 \wedge 1$

res^a[4] = 5^1^3

res^a[5] = $\cancel{5} \wedge 1 \wedge 3 \wedge \cancel{5} : 1 \wedge 3$

res^a[6] = $\cancel{1} \wedge 3 \wedge \cancel{1} = \underline{3}$.

}

return res;

→ logic.

time complexity = O(n)

Space Complexity = O(1)

Q2] Find the two non-repeating element in an array where every elements repeats twice

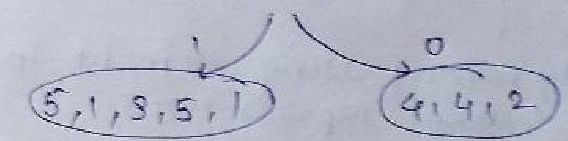
a = { 5, 4, 1, 4, 3, 5, 1, 2 }.

int res = 0;

in loop = { after performing XOR with each element of array & res we get

final res = 3^2.

Step 2: separate array in 2 parts such that right most bit of each no is 0 & 1



perform XOR operation

~~===~~ $\cancel{3}$^2^($\cancel{5}$,$\cancel{1}$,$\cancel{3}$,$\cancel{5}$,$\cancel{1}$) = 2 → a

Now 2^res = $\cancel{2}$^3^$\cancel{2}$ = 3 → b.

Time complexity = O(n)
Space complexity = O(1)

Q3] Find the only non-repeating element in an array where every other element repeats thrice.

a = { 2, 2, 1, 5, 1, 1, 2 }

solve it