Membership Operator

Membership operator is used to test whether a value or variable is found in a sequence (string , list , tuple , set etc.)

There are 2 membership operator.

- in
- not in

Behaviour of <u>in</u> and <u>not in</u>

in: The 'in' operator is used to check if a value exist in a sequence or not.

not in: The 'not in' operator return 'True' when value is not present.

Examples:-

a = 'Mayank'

('aya' in a) = True

('may' in a) = False

2 in [2, 3, 5, 7] = True

11 not in [1, 12, 3] = True

Binary / Bitwise Operators

A bit is most basic unit of information

Bit can be 0 or 1.

Bitwise operators work on bits.

So, integers have to be represented into binary form and then operation done on them.

Integer to binary representation

	2^3	2^2	2^1	2^0	
	8	4	2	1	
5		1	0	1	$_2$
6	0 0	1	1	0	
4		1	0	0	$_2$

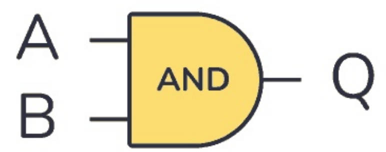
Bitwise operators

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR / Bitwise XOR
~	Bitwise inversion (one's complement)
<<	Shifts the bits to left / Bitwise Left Shift
>>	Shifts the bits to right / Bitwise Right Shift

Bitwise AND (&)

	4	2	1	
5	1	0	1	$_2$
4	1	0	0	$_2$
	1	0	0	

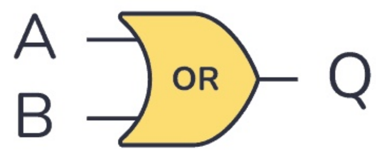
So the answer is 4.



A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Bitwise OR (|)

	4	2	1	
5	1	0	1	$_2$
4	1	0	0	$_2$
	1	0	1	



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

So the answer is 5.

Bitwise XOR (^)

	4	2	1
5	1	0	1_2
4	1	0	0_2
	0	0	1_2

So, the answer is 1.

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Left Shift (<<)

16 8 4 2 1

5 : 1 0 1 . 0 0

5<<1 : 1 0 1 0 = 10

5<<2 : 1 0 1 0 0 = 20

We shift the bits to the left

⇒ multiplying by 2

$$\boxed{a << n = a * 2^n}$$

Right shift (>>)

	4	2	1	
5:	1	0	1	
5 >> 1:		1	0	2
5 >> 2:			1	1

lost

We shift the bits to the right.

Bits can get lost if 1 is on right side

Bitwise Compliment (~)

Here the logic is little different due to the way <u>negative numbers are stored</u>. [bit on left side is used]

$$\begin{array}{ccccc} & 8 & 4 & 2 & 1 \\ 5 = & 0 & 1 & 0 & 1 \end{array}$$

$$\sim 5 = 1\ 0\ 1\ 0 = 10$$

Above is not 10 as we would have all 1 on left of 3rd bit

1's compliment = flipping the bits

2's Compliment = 1's compliment + 1

In machine, -ve number is stored as 2's Compliment.

-ve number

$\sim 5 = \underset{8}{1}\ \underset{4}{0}\ \underset{2}{1}\ \underset{1}{1} = \cancel{-3}$

Above is a -ve number but not -3 as 2's compliment is stored.

So what we want to do is

1. Take the result 1011
2. Take 1's compliment 0101
3. add 1 to it to get 2's compliment

$$
\begin{array}{r}
0101 \\
+\quad 1 \\
\hline
0110
\end{array} = 6
$$

8 4 2 1

$\Rightarrow$ answer is -6

*

$$
\begin{array}{r} 1 \\ +0 \\ \hline 1 \end{array} \qquad
\begin{array}{r} 0 \\ +0 \\ \hline 0 \end{array} \qquad
\begin{array}{r} 1 \\ +1 \\ \hline 10 \end{array}
$$

Shortcut is as follows

1. Take the number
2. Add 1 to it
3. Ans is result * -1

	8	4	2	1	
5 :	0	1	0	1	
	0	1	1	0	6

$\boxed{-6}$

$7 = 111$

$$\begin{array}{rrrrr} & 1 & 1 & 1 & \\ & 0 & 1 & 1 & 1 \\ + & & & & 1 \\ \hline & 1 & 0 & 0 & 0 \\ \hline & 8 & 4 & 2 & 1 \end{array}$$