BIT MANIPULATION BASIC QN's

☆ In this problem, we are given a number N and i. We have to check if i^{th} Bit in N's binary form is 1 or not.

Brute force solution is to obviously convert it to its binary and see what the i^{th} bit is.

Instead, we can left shift 1 i times and 'and' it.

Eg : N = 13   i = 2

1 >> 2

$(13)_{10} = (1101)_2 = (100)_2$

If we and them

```
      (i)
000····1101
000-···100
-----------
0····0x00
```

↳ Answer

So

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

Another Solution is to just right shift the original number i times to bring i^{th} bit to the end.

Then just check remainder of this number by 2, as that is our number.

Same process (via left shift) can can be followed but instead of AND we use OR.

return (N || (1 << i))

To clear the i^{th} bit, we take complement of 1 << i and 'and' it with N.

Eg : N = 13 i = 2

1 << 2

= $(100)_2$ $(13)_{10} = (1101)_2$

$\sim(100)_2 = (011)_2$

```
  0···1101
  1···1011
 00···1001
```

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

If the task was to toggle the ith bit. We will use the XOR gate, as XOR with 1 will just toggle it.

return (N ^ (1 << i))

We can also be asked to unset the last set bit in a number.
An observation is, when we subtract 1 from N, its last set bit changes to 0 and all the ones after it turn to 1.

Eg: N = 16 ⇒ N-1 = 15
10000 01111

N = 40 ⇒ N-1 = 39
101000 100111

So, just taking an and with N-1 will do.

Along similar lines, we can also figure out if a number is a power of 2 or not. Since any power of 2 will only have one set bit, 'and' with N-1 would mean a 0.

N = 32
10000
N = 31
01111
AND returns 0

return !(N & (N-1))

We can also be asked to count number of set bits in a binary numer. Bruteforce solution is to again go by normal method of converting the decimal to binary and checking all bits that were 1.

Another way to solve this problem is to just keep on toggling the last set bit until number turns zero.