You will be given a list of 32 bit unsigned integers. Flip all the bits ($1 \rightarrow 0$ and $0 \rightarrow 1$) and return the result as an unsigned integer.

Example

$n = 9_{10}$

$9_{10} = 1001_2$. We're working with 32 bits, so:

$00000000000000000000000000001001_2 = 9_{10}$

$11111111111111111111111111110110_2 = 4294967286_{10}$

Return 4294967286.

Function Description

Complete the flippingBits function in the editor below.

flippingBits has the following parameter(s):

- int n: an integer

Returns

- int: the unsigned decimal integer result

Input Format

The first line of the input contains q, the number of queries.

Each of the next q lines contain an integer, n, to process.

Constraints

$1 \leq q \leq 100$

$0 \leq n < 2^{32}$

Sample Input

```
3
2147483647
1
0
```

Sample Output

```
2147483648
4294967294
4294967295
```

Explanation

Take 1 for example, as unsigned 32-bits is 00000000000000000000000000000001 and doing the flipping we get 11111111111111111111111111111110 which in turn is 4294967294.

Change Theme Language Python 3

```python
#!/bin/python3

import math
import os
import random
import re
import sys

# Returns 32-bit binary number for a given decimal number
# Accepts an integer value
# Returns a binary number representation in the form of a list
def decimal_to_binary(number):
    if number == 0:
        return [0 for x in range(32)]

    stack = []
    while number > 0:
        stack.append(number % 2)
        number = number // 2

    binary = []
    while len(stack) > 0:
        binary.append(stack.pop())

    # conver to 32-bit representation
    binary_32_bit = [0 for x in range(32)]
    idx = len(binary_32_bit) - 1

    for jdx in reversed(range(len(binary))):
        binary_32_bit[idx] = binary[jdx]
        idx -= 1

    return binary_32_bit

# Accepts 32-bit binary number respresentation in the form of a list
# Returns the decimal number in the form of long integer
def binary_to_decimal(binary):
    decimal = 0
    power = 0
    for ele in reversed(binary):
        if ele == 1:
            decimal += 2**power
        power += 1
    return decimal

# Accepts 32-bit binary number respresentation in the form of a list
# Returns the binary number after flipping all ones with zeroes and vice versa
def flip_bits(binary):
    for idx, ele in enumerate(binary):
        if ele == 1: binary[idx] = 0
        if ele == 0: binary[idx] = 1
    return binary

# Complete the 'flippingBits' function below.
#
# The function is expected to return a LONG_INTEGER.
# The function accepts LONG_INTEGER n as parameter.
def flippingBits(n):
    # Write your code here
    return binary_to_decimal(flip_bits(decimal_to_binary(n)))




if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    q = int(input().strip())

    for q_itr in range(q):
```

Line: 57 Col: 52