Синтаксис Python. Битовые операции.

Любое целое число можно представить в двоичной системе счисления. В питоне это можно сделать при помощи функции *bin*(). Результатом функции *bin*() будет строка, начинающаяся с символов "0b". Например, print(bin(6)) напечатает 0b110. Обратное преобразование можно осуществить при помощи функции *int*():
>>>print(int('0b110', 2))
6
Если мы хотим преобразовать строку из нулей и единиц в число, нет необходимости писать "0b". Следующая программа будет работать точно также, как и предыдущая:
>>>print(int('110', 2))
Второй аргумент функции *int*() означает основание системы счисления. Например,
>>>print(int('110', 3))
12
На самом деле, неверно что для всех целых чисел в памяти хранится именно их бинарное представление. Для отрицательных чисел хранится так называемый *дополнительный код*. Но для положительных чисел хранится именно двоичное представление.
В питоне, как и в других языках программирования, есть битовые операции, которые работают напрямую с битами (единичками и ноликами, хранящимися в памяти):
& — побитовое И;
| — побитовое ИЛИ;
^ — побитовое ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR);
$\sim$ — побитовое ОТРИЦАНИЕ;
>> — побитовый СДВИГ ВПРАВО;
<< — побитовый СДВИГ ВЛЕВО.
Например,
>>>print(6 & 5) # 6 = 110, 5 = 101, 6 & 5 = 100
4
>>>print(6 | 5) # 6 = 110, 5 = 101, 6 | 5 = 111
7
>>>print(6 ^ 5) # 6 = 110, 5 = 101, 6 ^ 5 = 011
3
Обратите внимание, что
>>>print($\sim$ 6)
-7
Хотя, можно было бы предположить, что $6 = 110$, значит $\sim 6 = 001$. Такой эффект достигается из-за того, что целые числа в питоне имеют бит, отвечающий за знак, он инвертируется, а далее интерпритатор расшифровывает получившуюся последовательность бит уже из дополнительного кода, который используется для хранения отрицательных чисел. В итоге, $\sim N = -(N + 1)$. К слову, как устроен дополнительный код, мы разбирать не будем, если вам интересно, можете почитать википедию. Во всех задачах предполагается, что все числа положительные или 0, если не обговорено обратное.
Сдвиг влево сдвигает все биты влево на указанное количество позиций, при этом заполняет нулями освободившиеся биты. Сдвиг вправо сдвигает все биты вправо на указанное количество позиции, при этом несколько правых позиции отбрасываются. Например:
>>>print(9 >> 2) #$9 = 1001, 9 >> 2 = 10$
2
>>>print(5 << 1) #$5 = 101, 5 << 1 = 1010$
10
Все перечисленные выше операции можно использовать в комбинации со знаком =, так же как + = или $*$ =. Например,
>>>a = 5 #$a = 101$
>>>a <<= 1 #$a = 1010$
>>>a |= 6 #$a = 1010|110 = 1110$
>>>print(a)
14
Метод *int.bit_lenght*() возвращет количество бит, необходимых закодировать число. Другими словами, количество разрядов в двоичной записи числа. Например,

```
>>>a = 33
>>>print(a.bit_length())
6
```

A. *Неполное частное*
Дано два числа N и k. Выведите $\left[\frac{N}{2^k}\right]$. Можно использовать только битовые операции.

Input	Output
10 2	2

B. *Остаток*
Дано два числа N и k. Выведите остаток при делении N на 2^k. Помимо битовых операций можно вычитать 1.

Input	Output
37 3	5

C. *Инвертировать бит*
Дано два числа N и i. В числе N инвертируйте i-тый бит. То есть, если он был равен 0 сделайте его равным 1, если он был равен 1 сделайте его равным 0. Логические операции использовать запрещается.

Input	Output
15 2	11

D. *Без умножения умножьте*
Дано два числа N и M. Посчитайте их произведение, не использую операции умножения, деления, взятия остатка.

Input	Output
2 3	6

E. *Шифрование 1.*
Ваня решил зашифровать восьмибитное число, переставив все его биты влево по циклу: нулевой бит станет первым, первый — вторым, седьмой — нулевым. Помогите Ване.

Input	Output
129	3

F. *Шифрование 2.*
Юра придумал более сложный шифр: все биты сдвигаются влево по циклу на несколько позиций (не более чем 7, возможно на 0). Помогите Юре.

Input	Output
129 3	12

G. *Шифрование 3.*
Для шифрования 32-х битных чисел Коля и Максим решили поменять местами четные и нечетные биты. То есть, местами меняются 0-ой и 1-ый биты, 2-ой и 3-ий, и т.д. Использовать арифметические операции запрещается. Напишите программу, шифрующую числа таким образом.

Input	Output
78	141

H. *Единственный и неповторимый*
Насте говорят много целых чисел (неотрицательных). При этом известно, что все числа кроме одного были названы ровно по два раза, а это число только один раз. Необходимо найти это уникальное число. Учтите, что Настя не может запомнить все числа, которые ей говорят (в задаче запрещается использовать массивы).
На первой строке вводится одно нечетное число — количество чисел. Далее на каждой строке ровно по одному числу.

Input	Output
5 2 1 5 2 1	5

I. *Код, исправляющий ошибку*

В племени Мутумба всего N слов. Секретные службы племени закодировали каждое слово целым числом от 0 до $2^{32} - 1$ (включительно). Службы понимают, что во время передачи данных может возникнуть ошибка в нескольких битах. Поэтому, каждые два числа, участвующие в коде отличаются не менее чем в трех битах. Такой код называется *исправляющим одну ошибку.* Действительно, если во время передачи информации один бит будет передан ошибочно, принимающая сторона может найти число из кода, отличающееся от полученного ровно в одном бите. Такое число будет единственным.

На первой строчке через пробел вводятся N чисел, которыми закодированы слова племени Мутумба. Гарантируется, что любые два числа различаются не менее, чем в трех битах. На второй строчке число M, которое необходимо расшифровать. То есть, если M отличается от одного из чисел, которым зашифровано слово, не более, чем в одном бите, необходимо вывести число, которое нам хотели передать. Если M отличается от каждого числа не менее, чем в двух битах, выведите "Ne ponimat'"(двойные кавычки выводить не нужно).

Input	Output
537966573 2845156959 1442095319	Ne ponimat'

Input	Output
1 209 47 43	47