Задача 3. XOR double

Источник:	базовая
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Требуется взять заданное число X типа `double`, "прохогить" его с заданным 64-битным целым числом M, и вывести результат как `double`. "Прохогить" означает: обратить в битовом представлении X все биты, для которых бит с тем же номером в битовом представлении M равен единице.

В этой задаче предполагается little-endian порядок байтов и общепринятое 8-байтовое представление `double`.

Формат входных данных

В первой строке записано целое число N — количество тестов ($1 \leqslant N \leqslant 1\,000$). В остальных N строках записаны тесты, по одному в строке.

Каждый тест описан в формате: `"P/Q xor M"`. Здесь целые числа `P` и `Q` — числитель и знаменатель дроби, задающей вещественное число X ($0 \leqslant P \leqslant 100$, $1 \leqslant Q \leqslant 100$), а `M` — шестнадцатеричное целое число M ровно из шестнадцати цифр. В записи `M` сначала идут старшие цифры, потом младшие (как обычно у людей записываются числа).

Совет: Шестнадцитеричное число можно читать при помощи формата `"%x"` так же, как мы считаем десятичные числа форматом `"%d"`. Если нужно прочитать 64-битное число, то нужно дописать две буквы `ll` перед последней буквой формата.

Формат выходных данных

Для каждого теста выведите в отдельной строке (X xor M) как вещественное число типа `double`.

Ваши ответы должны быть верны с относительной точностью 10^{-14}. Рекомендуется использовать формат `"%0.15g"` при выводе ответа.

Пример

input.txt	output.txt
10	-0
0/1 xor 8000000000000000	1
1/1 xor 0000000000000000	-1
1/1 xor 8000000000000000	0
1/1 xor 3ff0000000000000	2
1/1 xor 7ff0000000000000	0.5
1/1 xor 0010000000000000	1.625
1/1 xor 000a000000000000	-0.428571428571429
3/7 xor 8000000000000000	-2.90689205178751e-054
3/7 xor 8b0abc0000000000	0.428571428570292
3/7 xor 000000000000d000	

Пояснение к примеру

В первом тесте $X = 0/1$, а в заданной маске M установлен только старший бит. В представлении нуля в `double` все биты нулевые, после операции xor старший бит становится

единичным, однако число по-прежнему остаётся нулевым (получается так называемый “отрицательный ноль”).

Во втором тесте число $X = 1/1$ равно единице, а маска M вся нулевая. Значит xor ничего не меняет и результат получается тоже равен единице.

В третьем и восьмом тестах в маске только старший бит единичный. Он в представлении `double` отвечает за знак числа, так что в этих тестах число X меняет знак.

В предпоследнем тесте число $X = 3/7$, его битовое представление выглядит как `3fdb6db6db6db6db` в шестнадцатеричном виде. Когда мы xor-им с заданной маской, получается представление `b4d1d1b6db6db6db`. Если проинтерпретировать эти данные как `double`, то получается число `-2.90689205178751e-054`.