Задача A. Угадайте сумму

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Это **интерактивная** задача.

Вам задается целое положительное число n и целые числа l и r, такие, что $0 \leqslant l \leqslant r < 2^n$. У жюри загадана последовательность a длины 2^n $a_0, a_1, \ldots, a_{2^n-1}$, состоящая из целых чисел от 0 до 99 включительно.

Вам необходимо найти сумму чисел на отрезке с l-го по r-й по модулю 100. Однако вы не можете напрямую узнать значения элементов в последовательности a. Вместо этого вы можете делать следующие запросы:

Выберите неотрицательные целые числа i и j такие, что $2^i(j+1) \leqslant 2^n$. Пусть $L = 2^i j$ $R = 2^i(j+1) - 1$. Тогда жюри вернет сумму чисел на отрезке с L-го по R-й по модулю 100.

Пусть m — минимальное количество вопросов, необходимое для определения суммы чисел на отрезке с l-го по r-й для любой последовательности a. Вам нужно найти эту сумму за m запросов.

Протокол взаимодействия

При запуске решения на вход вашей программе подается три целых числа n, l, r ($1 \leqslant n \leqslant 18$, $0 \leqslant l \leqslant r \leqslant 2^n - 1$) — показатель степени для размера массива и границы отрезка.

После этого вы можете выводить запросы «`? i j`», где $i, j \geqslant 0$, $2^i(j+1) \leqslant 2^n$. Интерактор ответит вам одно целое число t — сумма чисел на на отрезке с L-го по R-й по модулю 100.

В тот момент, когда вы решили, что знаете сумму на нужном отрезке, выведите ответ в формате «`! s`». Обратите внимание, вы не можете сделать больше чем m запросов.

Пример

стандартный ввод	стандартный вывод
3 1 5 41 85 11	? 0 1 ? 1 1 ? 1 2 ! 37

Задача B. Тензор

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Это интерактивная задача.

Вам дано целое число n.

Жюри загадало ориентированный граф с n вершинами (пронумерованными от 1 до n) и с некоторым количеством рёбер. Дополнительно известно, что:

- Граф содержит рёбра только вида $i \leftarrow j$, где $1 \leqslant i < j \leqslant n$.
- Для любых трёх вершин $1 \leqslant i < j < k \leqslant n$ верно хотя бы одно из следующего[†]:
  - Вершина i достижима из вершины j, или
  - Вершина i достижима из вершины k, или
  - Вершина j достижима из вершины k.

Вы хотите покрасить каждую вершину графа либо в белый, либо в чёрный цвет так, чтобы для любых двух вершин i и j $(1 \leqslant i < j \leqslant n)$ одного цвета вершина i была достижима из вершины j.

Для этого вы можете делать запросы следующего вида:

- `? i j` — достижима ли вершина i из вершины j $(1 \leqslant i < j \leqslant n)$?

Найдите любую подходящую раскраску вершин графа за не более чем $2 \cdot n$ запросов. Можно доказать, что такая раскраска всегда существует.

Примечание: интерактор **не** является адаптивным, то есть граф фиксируется до выполнения каких-либо запросов.

[†] Вершина a достижима из вершины b если в графе существует путь из вершины b в вершину a.

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. В первой строке находится одно целое число t $(1 \leqslant t \leqslant 1000)$ — количество наборов входных данных. Далее следует описание наборов входных данных.

Единственная строка каждого набора входных данных содержит единственное целое число n $(3 \leqslant n \leqslant 100)$ — количество вершин в графе.

Гарантируется, что сумма n по всем наборам входных данных не превышает 1000.

Протокол взаимодействия

Взаимодействие для каждого набора входных данных начинается с чтения целого числа n.

Для формирования запроса выведите «`? i j`» без кавычек $(1 \leqslant i < j \leqslant n)$. Если вершина i достижима из вершины j, вы получите в ответ `YES`. Иначе, вы получите в ответ `NO`.

Если вместо ответа или допустимого значения n вы получите целое число -1, то это означает, что ваша программа сделала некорректный запрос, превысила лимит запросов или дала неверный ответ на предыдущем наборе входных данных. При получении вердикта «Неправильный ответ» ваша программа должна немедленно завершиться. В противном случае можно получить произвольный вердикт, поскольку решение будет продолжать считывать данные из закрытого потока.

Когда вы будете готовы дать окончательный ответ, выведите «`!` c_1 c_2 ... c_n» без кавычек — раскраска вершин графа, где $c_i = 0$, если вершина чёрная, и $c_i = 1$, если вершина белая. После решения одного набора входных данных программа должна сразу же переходить к следующему. После решения всех наборов входных данных программа должна быть немедленно завершена.

После вывода запроса не забудьте вывести перевод строки и сбросить буфер вывода. В противном случае вы получите вердикт `Решение «зависло»`. Для сброса буфера используйте:

- `fflush(stdout)` или `cout.flush()` в C++;

- `System.out.flush()` в Java;
- `flush(output)` в Pascal;
- `stdout.flush()` в Python;
- смотрите документацию для других языков.

Взломы

Для взлома используйте следующий формат:

Первая строка содержит целое число t ($1 \leqslant t \leqslant 1000$) — количество наборов входных данных.

Первая строка каждого набора входных данных содержит два целых числа n и m ($3 \leqslant n \leqslant 100$, $0 \leqslant m \leqslant \frac{n \cdot (n-1)}{2}$) — количество вершин и рёбер в графе.

Следующие m строк должны содержать два числа a и b ($1 \leqslant b < a \leqslant n$), означающие, что в графе присутствует ребро $a \to b$. Граф должен удовлетворять условиям выше.

Сумма n по всем наборам входных данных не должна превышать 1000.

Пример

стандартный ввод	стандартный вывод
2 4 YES YES YES NO NO NO 5	? 1 2 ? 2 3 ? 1 3 ? 1 4 ? 2 4 ? 3 4 ! 0 0 0 1 ! 1 1 0 1 0

Замечание

Взаимодействие во втором примере происходит следующим образом:

Решение	Жюри	Объяснение
	2	Есть 2 набора входных данных.
	4	В первом наборе входных данных загадан граф с 4-мя вершинами.
? 1 2	YES	Решение спрашивает, достижима ли вершина 1 из вершины 2, жюри отвечает YES.
? 2 3	YES	Решение спрашивает, достижима ли вершина 2 из вершины 3, жюри отвечает YES.
? 1 3	YES	Решение спрашивает, достижима ли вершина 1 из вершины 3, жюри отвечает YES.
? 1 4	NO	Решение спрашивает, достижима ли вершина 1 из вершины 4, жюри отвечает NO.
? 2 4	NO	Решение спрашивает, достижима ли вершина 2 из вершины 4, жюри отвечает NO.
? 3 4	NO	Решение спрашивает, достижима ли вершина 3 из вершины 4, жюри отвечает NO.
! 0 0 0 1		Решение каким-то образом определило, что такая раскраска подходит, и выводит её. Поскольку ответ правильный, жюри переходит к следующему набору входных данных.
	1	Во втором наборе входных данных загадан граф с 5-ю вершинами.

Обратите внимание, что пустые строки во входных и выходных данных примера сделаны для наглядности и не встречаются в реальном взаимодействии.

Задача C. Conv19

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Это интерактивная задача.

В свете недавних событий было решено отменить чемпионат мира по шахматам. Гроссмейстеры, сдав обратно авиабилеты, начали думать, чем бы занять свое освободившееся время. Они наткнулись на шахматную онлайн игру «Conv19». В этой игре есть шахматное поле $N \times N$ и всего одна фигура — клетка вируса. Эта клетка бьет некоторые фигуры следующим образом: перед началом игры фиксируется нечетное число $M \geqslant 3$; берется черно-белая шахматная раскраска квадрата 5×5, где центр является клеткой черного цвета; после этого каждая клетка такой раскраски меняется на квадрат из $M \times M$ клеток. Если вирус находился в клетке (x, y), то множество клеток, которые находились под его ударом, определялось так: центр получившегося квадрата размером $5M \times 5M$ прикладывался к точке (x, y), и все черные клетки считались клетками под ударом. При этом считается, что вирус мог находиться только в тех клетках поля, куда можно было приложить весь квадрат (то есть, сделать это так, чтобы он не вылез за границы поля).

Процесс игры выглядит так. Один игрок кладет вирус в произвольную клетку поля. После этого ходы делает только второй игрок. За ход он может узнать про произвольную клетку — бьет ее вирус или нет. Цель второго игрока — отгадать, где находится вирус, за не более, чем 300 ходов.

Костя решил сыграть в эту игру с Ваней. Поскольку на большом поле игра может затянуться надолго, Костя решил дать Ване подсказку, и указал ему на точку, которая бьется вирусом. Ваня решил, что готовить контест важнее, чем играть в какие-то игры, поэтому попросил вас написать программу, которая выиграет за него. Обратите внимание, что число M Костя выберет сам, и не скажет его Ване.

Протокол взаимодействия

В начале выполнения вашей программе на вход подается число N ($15 \leqslant N \leqslant 2\,000\,000\,000$), а также координаты клетки, которую бьет вирус X, Y ($1 \leqslant X \leqslant N$, $1 \leqslant Y \leqslant N$).

После этого вы можете выводить запросы «examine x y», где $1 \leqslant x, y \leqslant N$, чтобы узнать, бьет ли вирус клетку (x, y). Интерактор ответит вам «true» или «false», соответствующие ответу на этот вопрос.

В тот момент, когда вы посчитали, что знаете клетку с вирусом, выведите ответ в формате «solution x y».

Пример

стандартный ввод	стандартный вывод
20 4 9 false true false true true false true false true	examine 2 9 examine 3 9 examine 6 9 examine 5 9 examine 4 3 examine 2 3 examine 3 3 examine 3 1 examine 3 2 solution 10 9

Замечание

Соблюдайте формат ввода-вывода, не забывайте очищать за собой буфер, и не болейте.

Задача D. Поиграем?

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Это **интерактивная** задача.

1804 год. Вице-президент Соединённых Штатов Аарон Бёрр вызывает на дуэль кандидата в губернаторы штата Нью-Йорк Александра Гамильтона за серию оскорбительных памфлетов в свой адрес.

Но Бёрр разумный человек и понимает, что даже если он убьет Гамильтона на дуэли, он потеряет свою репутацию, и его политическая карьера будет закончена. Поэтому противники решили просто сыграть в игру. Для честности они решили сыграть в неё g раз.

Каждую игру Гамильтон загадывает целое положительное число n, а Бёрр пытается его отгадать. Для любого целого положительного x Бёрр может спросить у Гамильтона долю чисел между 1 и n включительно, которые делятся на x. Иными словами, спрашивая про x, он получает значение выражения

$$\frac{\left\lfloor \frac{n}{x} \right\rfloor}{n},$$

причём Гамильтон сообщает ему результат в виде **несократимой** дроби (здесь $\lfloor r \rfloor$ обозначает результат округления вниз вещественного числа r).

Помогите Бёрру найти ответ за некоторое заранее определённое число запросов.

Протокол взаимодействия

При запуске решения на вход подается целое число g — число игр между Гамильтоном и Бёрром ($1 \leqslant g \leqslant 1000$).

Для каждого теста зафиксировано число q ($6 \leqslant q \leqslant 60$) — максимальное количество запросов в одной игре. Гарантируется, что q запросов достаточно, чтобы решить задачу. Эти числа не сообщаются программе участника, но ограничения на эти числа в различных подзадачах приведены в таблице системы оценивания. Если программа участника делает более q запросов программе жюри, на этом тесте она получает в качестве результата тестирования «Wrong answer».

Чтобы сделать запрос, следует вывести строку «`X` t», где t — целое положительное число ($1 \leqslant t \leqslant 10^{18}$), для которого требуется узнать значение выражения

$$\frac{\left\lfloor \frac{n}{t} \right\rfloor}{n}$$

В ответ на каждый запрос программа получает через пробел два числа a и b — числитель и знаменатель этой дроби **после сокращения** — или число -1 в случае, если программа превысила ограничение по числу запросов.

Если программа определила загаданное число, то она должна вывести строку «`N` t», где t — предполагаемый ответ ($1 \leqslant t \leqslant 10^{18}$). Если ответ был правильный, то в ответ программа получает строку «`Correct`», а если неправильный, то она получает строку «`Wrong`».

После этого программа переходит к следующей игре, если они остались, иначе она должна завершиться.

Обратите внимание, в случае считывания числа -1 или строки «`Wrong`» вы **обязательно** должны сразу завершить вашу программу. В противном случае вердикт тестирующей системы может быть некорректным, в частности, вы можете получить вердикт Run-time error или Time limit exceeded!

Гарантируется, что в каждом тесте загаданные числа фиксированы в самом начале игры и не изменяются в зависимости от ваших запросов.

Система оценки

Тесты к этой задаче состоят из девяти групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов **предыдущих** групп.

Группа	Баллы	Дополнительные ограничения
0	0	$q = 60$
1	30	$q = 60$
2	10	$q = 30$
3	4	$q = 20$
4	4	$q = 15$
5	5	$q = 10$
6	5	$q = 9$
7	11	$q = 8$
8	10	$q = 7$
9	21	$q = 6$

Пример

стандартный ввод	стандартный вывод
2 1 2 3 10 1 5 1 5 1 10 1 10 0 1 Correct 1 1 0 1 Correct	 X 2 X 3 X 5 X 4 X 6 X 10 X 11 N 10 X 1 X 2 N 1

Замечание

В первом примере $g = 2$. Приведены примеры запросов, по которым игрок угадывает, что в первой игре загадано число 10, а во второй 1. Эти же числа загаданы в первом тесте в тестирующей системе.

В точности соблюдайте формат выходных данных. После каждого вывода обязательно выводите один перевод строки и сбрасывайте буфер вывода — для этого используйте `flush(output)` на языке Паскаль или Delphi, `fflush(stdout)` или `cout.flush()` в C/C++, `sys.stdout.flush()` на языке Python, `System.out.flush()` на языке Java.

Задача E. Новогодний и прямоугольный

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Это **интерактивная** задача.

На Новый год Дед Мороз подарил Глебу то, о чём он уже давно мечтал — клетчатый квадрат размером $n \times n$. Подарок этот не простой, а с сюрпризом — внутри квадрата Дед Мороз выбрал некоторый непустой прямоугольник, и в каждую клетку этого прямоугольника он положил по мандарину.

Теперь, чтобы получить желанный подарок, Глебу нужно сыграть с Дедом Морозом в очень интересную игру. Глеб должен отгадать, в каком именно прямоугольнике находятся все мандаринки, подаренные Дедом Морозом. Будем считать, что строки и столбцы занумерованы числами от 1 до n снизу вверх и слева направо. Глеб может производить два типа запросов:

- ? $x_1\, y_1\, x_2\, y_2$ ($1 \leqslant x_1 \leqslant x_2 \leqslant n$, $1 \leqslant y_1 \leqslant y_2 \leqslant n$) — в ответ на этот запрос Дед Мороз говорит, сколько мандаринок находится в прямоугольнике, левым нижним углом которого является клетка (x_1, y_1), а правым верхним — клетка (x_2, y_2);
- ! $x_1\, y_1\, x_2\, y_2$ ($1 \leqslant x_1 \leqslant x_2 \leqslant n$, $1 \leqslant y_1 \leqslant y_2 \leqslant n$) — когда Глеб уверен, что он точно знает, где находятся мандаринки, он должен сделать запрос такого вида, чтобы сообщить свой ответ. При этом (x_1, y_1) соответствует предполагаемому расположению левого нижнего угла, а (x_2, y_2) — правого верхнего.

Формат входных данных

При запуске решения на вход вашей программе подается одно число n ($1 \leqslant n \leqslant 2 \cdot 10^9$) — размер квадрата.

Затем на каждый запрос типа "`?`" вам будет выдаваться количество мандаринок, находящихся в указанном вами прямоугольнике.

Формат выходных данных

Вы должны выводить корректные запросы в формате, описанном выше. Последним должен следовать единственный запрос вида "`!`", после чего ваша программа должна немедленно завершиться. Ваша программа должна произвести не больше q (параметр зависит от номера группы) запросов типа "`?`". Обратите внимание, что последний запрос, выводящий ответ, не входит в данные q запросов.

В точности соблюдайте формат выходных данных. После вывода каждой строки сбрасывайте буфер вывода — для этого используйте команды `flush(output)` на языке Паскаль или Delphi, `fflush(stdout)` или `cout.flush()` в C/C++, `sys.stdout.flush()` на языке Python, `System.out.flush()` на языке Java.

Примеры

стандартный ввод	стандартный вывод
4 6 6 4 	? 1 1 4 4 ? 1 3 4 4 ? 2 3 4 4 ! 1 3 3 4

Замечание

Пример в условии иллюстрирует взаимодействие с проверяющей программой. Для прохождения первого теста не обязательно производить такие же запросы, как в примере.

Тесты к этой задаче состоят из шести групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов **предыдущих** групп.

Группа	Тесты	Баллы	Дополнительные ограничения		Комментарий
			n	q	
0	1 – 1	0	$n = 4$	$q = 1000$	Тест из условия.
1	2 – 12	10	$n \leqslant 10$	$q = 10\,000$	
2	13 – 23	20	$n \leqslant 100$	$q = 10\,000$	
3	24 – 34	20	$n \leqslant 10\,000$	$q = 20\,000$	
4	35 – 45	25	$n \leqslant 2 \cdot 10^9$	$q = 128$	
5	46 – ∞	25	$n \leqslant 2 \cdot 10^9$	$q = 64$	

Задача F. ПРС и криминальный список (простая версия)

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Это простая версия задачи. Единственное различие между простой и сложной версиями заключается в том, что в этой версии $k = 2$. Вы можете делать взломы, только если решили обе версии этой задачи.

Это интерактивная задача.

Для каждого десятичного числа есть соответствующая ему k-ичная запись. Цифры числа, записанного в k-ичной системе счисления, будем называть k-цифрами. Определим k-ичный XOR двух k-цифр a и b как $(a + b) \bmod k$.

k-ичный XOR двух k-ичных чисел равен числу, полученному попарным k-ичным XOR соответствующих k-цифр. k-ичный XOR двух десятичных чисел a и b обозначается $a \oplus_k b$ и равен десятичному представлению k-ичного XOR a и b, записанных в k-ичной системе счисления. Все числа, встречающиеся далее в условии, записаны в десятичной системе счисления, если не указано иное. В случае $k = 2$, что всегда верно для этой версии задачи, k-ичный XOR — это то же самое, что побитовое исключающее ИЛИ.

Вы взломали базу преступлений Полиции Рокпорт-Сити (ПРС), так же известную как Криминальный Список. Но, чтобы получить к ней доступ, вам нужен пароль. Вы его не знаете, но вполне уверены, что его значение лежит между 0 и $n-1$ включительно. Вы решили угадать его. Вы можете сделать не более чем n попыток, далее система заблокируется. Система адаптивна – каждый раз, когда вы делаете неверную попытку, она меняет пароль. А именно, если до попытки пароль был равен x и вы попытались ввести другое число y, то система поменяет пароль на число z такое, что $x \oplus_k z = y$. Угадайте пароль и взломайте систему.

Формат входных данных

Первая строка содержит одно целое число t ($1 \leqslant t \leqslant 10\,000$) — количество наборов входных данных. Далее следует описание t наборов входных данных.

Первая строка каждого набора входных данных содержит два целых числа n ($1 \leqslant n \leqslant 2 \cdot 10^5$) и k ($k = 2$).

Гарантируется, что сумма n по всем наборам входных данных не превосходит $2 \cdot 10^5$.

Протокол взаимодействия

Для каждого набора входных данных считайте два целых числа n и k. Вам разрешено сделать не более n попыток угадать пароль.

Для каждой попытки выведите одно целое число y ($0 \leqslant y \leqslant 2 \cdot 10^7$). Пусть текущий пароль равен x. Считайте одно целое число r.

Если $x = y$, вы считаете $r = 1$, и этот набор входных данных считается решенным. Далее вы должны продолжить решать оставшиеся наборы.

Иначе вы считаете $r = 0$, и система поменяет пароль на число z такое, что $x \oplus_k z = y$.

После каждой попытки не забудьте вывести перевод строки и сбросить буфер вывода. В противном случае вы получите вердикт `Решение «зависло»`. Для сброса буфера используйте:

- `fflush(stdout)` или `cout.flush()` в C++;
- `System.out.flush()` в Java;
- `flush(output)` в Pascal;
- `stdout.flush()` в Python;
- смотрите документацию для других языков.

Если вы сделаете некорректную попытку или превысите лимит в n попыток, вы считаете $r = -1$ вместо ответа и получите вердикт `Неправильный ответ`. В таком случае ваша программа должна немедленно завершается, чтобы избежать неопределенных вердиктов.

Обратите внимание, что система **адаптивна**. Это значит, что изначальный пароль не зафиксирован в начале и может зависеть от ваших запросов. Гарантируется, что в любой момент времени существует хотя бы один начальный пароль, для которого согласуются все ответы на запросы.

Взломы:

Для взломов используйте следующий формат:

Первая строка должна содержать одно целое число t ($1 \leqslant t \leqslant 10\,000$) — количество наборов входных данных.

Первая и единственная строка каждого набора должна содержать два целых числа n ($1 \leqslant n \leqslant 2 \cdot 10^5$) и k ($k = 2$), равные соответственно количеству возможных попыток и основанию системы счисления. Оптимальный начальный пароль будет автоматически подобран системой.

Сумма n по всем наборам входных данных не должна превышать $2 \cdot 10^5$.

Пример

стандартный ввод	стандартный вывод
1 5 2 0 0 1	3 4 5

Замечание

В этом наборе входных данных загаданный пароль равен 2.

Первая попытка равна 3. Это не равно текущему паролю. Поэтому система возвращает ответ 0 и меняет пароль на 1, поскольку $2 \oplus_2 1 = 3$.

Вторая попытка равна 4. Это не равно текущему паролю. Поэтому система возвращает ответ 0 и меняет пароль на 5, поскольку $1 \oplus_2 5 = 4$.

Третья попытка равна 5. Это равно текущему паролю. Поэтому система возвращает ответ 1 и работа сделана.

Заметьте, что начальный пароль был взят таким для наглядности объяснения. В действительности система может повести себя иначе, поскольку она адаптивна.

Задача G. Угадать количество делителей

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Это интерактивная задача.

Загадано число $1 \leqslant X \leqslant 10^9$. Вам **не нужно** угадывать это число. Вам нужно **определить количество делителей** этого числа, и даже это вам **не нужно делать точно**: ваш ответ будет считаться верным, если его абсолютная погрешность не превышает 7 **или** его относительная погрешность не превышает 0.5. Формально, пусть ваш ответ равен ans, а количество делителей X равно d, тогда ваш ответ будет считаться правильным, если выполнено **хотя бы одно** из следующих двух условий:

- $|ans - d| \leqslant 7$
- $\frac{1}{2} \leqslant \frac{ans}{d} \leqslant 2$

Вы можете не более 22 раз сделать запрос. Запрос состоит из одного числа $1 \leqslant Q \leqslant 10^{18}$. В ответ на запрос вы получите $gcd(X, Q)$ — наибольший общий делитель X и Q.

Число X зафиксировано до всех запросов. Иными словами, **интерактор не является адаптивным**.

Назовём процесс отгадывания количества делителей числа X *игрой*. В рамках одного теста вам нужно будет сыграть T независимых игр, то есть отгадать количество делителей T раз для T независимых чисел X.

Формат входных данных

На первой строке записано одно целое число T ($1 \leqslant T \leqslant 100$) — количество игр.

Протокол взаимодействия

Чтобы сделать запрос, выведите строку вида «`? Q`» ($1 \leqslant Q \leqslant 10^{18}$). После запроса считайте одно число — $gcd(X, Q)$. Вы можете сделать не более 22 таких запросов в рамках одной игры.

Если вы считаете, что знаете количество делителей X с достаточной точностью, выведите ваш ответ в формате «`! ans`». ans должно быть целым числом. Если это последняя игра, то вы должны завершить выполнение программы, иначе вы должны начать следующую игру. Обратите внимание, что интерактор не выводит ничего в ответ на вывод ответа.

После вывода запроса или ответа не забудьте вывести перевод строки и сбросить буфер вывода. Для сброса буфера вывода используйте:

- `fflush(stdout)` или `cout.flush()` в C++;
- `System.out.flush()` в Java;
- `flush(output)` в Pascal;
- `stdout.flush()` в Python;
- смотрите документацию для других языков.

Пример

стандартный ввод	стандартный вывод
2 1 1 1 1024 1048576 4194304	 ? 982306799268821872 ? 230856864650023977 ? 134690134760714371 ! 5 ? 1024 ? 1048576 ? 1073741824 ! 42

Замечание

Почему ограничение на запросы именно 22? Возможно, автор задачи — фанат Тейлор Свифт.

Рассмотрим пример из условия.

В первой игре загадано число $X = 998\,244\,353$. Было бы сложно это угадать, правда? Это число является простым, то есть количество его делителей равно 2. Решение сделало запросы с несколькими случайными числами, и ответы на все запросы оказались равны 1 (удивительно, что ни один из трёх запросов не оказался кратным $998\,244\,353$). Логично предположить, что у загаданного числа не очень много делителей, поэтому решение ответило 5. Почему бы и не 5. Этот ответ будет засчитан, так как $|5 - 2| = 3 \leqslant 7$.

Во второй игре загадано число $X = 4\,194\,304 = 2^{22}$, количество его делителей равно 23. Решение сделало запросы $1024 = 2^{10}$, $1\,048\,576 = 2^{20}$, $1\,073\,741\,824 = 2^{30}$ и получило ответы $1024 = 2^{10}$, $1\,048\,576 = 2^{20}$, $4\,194\,304 = 2^{22}$, соответственно. Затем решение окончательно запуталось и выдало ответ на Главный вопрос жизни, Вселенной и всего такого. Этот ответ будет засчитан, так как $\frac{1}{2} \leqslant \frac{42}{23} \leqslant 2$.

Задача H. Архивы джедаев

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Это интерактивная задача.

В Далёкой-Далёкой Галактике 10^{18} планет, и все они пронумерованы натуральными числами от 1 до 10^{18}.

В Архивах джедаев были сохранены сведения о всех планетах Галактики: по кругу в огромном зале были расположены ячейки, пронумерованные против часовой стрелки от 1 до 10^{18}. Исходно Архивы были устроены очень просто: в ячейке с номером i хранились сведения о планете с номером i. Поэтому, зная лишь номер планеты, всегда можно было легко найти информацию о ней.

Вот и Оби-Вану Кеноби понадобилось узнать координаты планеты Камино, которая имеет номер x. Однако, заглянув в Архивы, он с удивлением обнаружил, что кто-то удалил из Архивов m ячеек со сведениями о некоторых планетах. Кроме того, оставшиеся ячейки были заново пронумерованы, начиная с 1, против часовой стрелки (возможно, начиная не с той, с которой они нумеровались исходно), и теперь в ячейке с номером i могут храниться сведения о совсем другой планете.

Оби-Ван в затруднении. Ему нужно срочно найти данные о планете Камино. Оби-Ван может проверить содержимое ячейки и узнать, данные о какой планете в ней находятся. У Оби-Вана мало времени, поэтому он может проверить содержимое не более чем десяти ячеек.

Помогите ему выяснить, в какой ячейке находятся данные о планете Камино, если, конечно, они не были удалены из Архивов.

Протокол взаимодействия

Сначала вашей программе подаётся на вход в отдельной строке два числа: x — номер планеты Камино ($1 \leqslant x \leqslant 10^{18}$) и m — число удаленных из Архивов ячеек ($0 \leqslant m \leqslant 500$).

После этого ваша программа может делать запросы: «посмотреть, сведения о какой планете записаны в ячейке номер v». Для этого нужно вывести в выходной поток на отдельной строке «`? v`» ($1 \leqslant v \leqslant 10^{18} - m$). В ответ на запрос во входном потоке на отдельной строке будет записано одно число — номер планеты, сведения о которой записаны в ячейке v. Вы можете сделать не более десяти таких запросов, иначе ваша программа получит вердикт «`Wrong answer`».

В конце вы должны вывести в выходной поток «`! a`», где a — это номер ячейки, в которой записаны сведения о планете Камино. Если данные о планете Камино были удалены из Архива, то следует вместо номера ячейки вывести -1, таким образом последнее сообщение должно быть «`! -1`».

Пример

стандартный ввод	стандартный вывод
16 3 10 12 13 16	? 1 ? 2 ? 3 ? 4 ! 4

Замечание

В примере из Архива были удалены данные о планетах с номерами 11, 14 и 15, а ячейки были перенумерованы, начиная с ячейки, содержащей сведения о планете с номером 10.

Задача I. Инверсия и Разворот

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Это интерактивная задача.

Вы играете со своим другом в крайне увлекательную игру. Ваш друг, будучи программистом, загадал строку, состоящую из символов «0» и «1» длины B.

Вы хотите отгадать эту строку. Для этого вы можете назвать число P ($1 \leqslant P \leqslant B$), после чего ваш друг сообщит P-й символ загаданной строки.

Спустя несколько часов не очень увлекательной игры ваш друг предложил усложнить игру. Пусть вы хотите спросить у вашего друга очередной символ загаданной строки, причем это уже Q-й ваш вопрос. Если число Q оканчивается на цифру «1», то есть равно $1, 11, 21, 31, 41, \ldots$, то ваш друг, прежде чем ответить на ваш вопрос, случайно равновероятно выбирает число от 1 до 4 и выполняет соответствующую операцию:

1. Инвертировать все биты в строке, то есть заменить все символы «0» на «1» и наоборот.
2. Развернуть строку, то есть поменять местами первый и последний символы, второй и предпоследний символы, и так далее.
3. Инвертировать, а затем развернуть строку.
4. Ничего не делать.

После того, как друг выполнил одну из операций, он сообщает вам P-й символ получившейся строки.

Теперь, так как правила игры изменились, вы хотите научиться выигрывать, а именно: используя не более 150 вопросов отгадать строку.

Обратите внимание, вы должны отгадать не изначальную строку, а строку после всех сделанных изменений.

Протокол взаимодействия

Для начала считайте два числа T и B ($1 \leqslant T \leqslant 100$) — количество тестовых случаев и длину загаданной строки.

После этого для каждого тестового случая повторяется следующая процедура. Во всех тестовых случаях длина строки одинаковая, однако сами загаданные строки могут различаться.

Вы можете задавать программе жюри вопросы следующего вида: `?` P. После того, как вы задали вопрос, происходят следующие действия: если ваш вопрос является 1-м, 11-м, 21-м, и так далее, то сначала программа жюри выполняет одну из четырех описанных операций. Номер операции выбирается случайно равновероятно. После этого программа жюри выдаст в качестве ответа символ «0» или «1» (без кавычек).

После того, как вы считаете, что готовы сделать ответ, выведите ответ в формате: `!` S, где S — строка длины B, состоящая из символов «0» и «1». После того, как вы вывели ответ, программа жюри выдаст в качестве ответа символ «Y», если ваш ответ является правильным, либо «N» в противном случае (символы выдаются без кавычек). Если ваш ответ не является правильным, вы должны немедленно завершить программу.

Количество вопросов для каждого тестового случая не должно превосходить 150. В случае несоблюдения описанного протокола взаимодействия с программой жюри, вы можете получить произвольный вердикт.

Обратите внимание, что каждое выведенное вами сообщение должно завершаться переводом строки. Также после вывода каждого сообщения ваша программа должна очищать потоковый буфер, чтобы выведенная вами информация дошла до программы жюри: например, это делают вызовы `fflush(stdout)` или `cout.flush()` в C++, `System.out.flush()` в Java, `Console.Out.Flush()` в C#, `flush(output)` в Pascal, `sys.stdout.flush()` в Python.

Система оценки

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Дополнительные ограничения	Оценка	Необходимые подзадачи
0	0	Тесты из условия	подзадача	–
1	10	$B = 10$	подзадача	–
2	40	$B = 20$	подзадача	–
3	50	$B = 100$	подзадача	1

Пример

стандартный ввод	стандартный вывод
1 12 0 0 0 1 Y	 ? 1 ? 2 ? 3 ? 12 ! 000101110101

Замечание

Пример лишь иллюстрирует протокол взаимодействия. Не гарантируется, что ответ на него действительно такой. Переводы строк в примере выполнены для удобства.

Для начала узнаем первый символ загаданной строки. Так как это первый вопрос, его номер заканчивается на цифру «1», поэтому со строкой произойдет одна из четырех операций, после чего программа жюри сообщает, что первый символ равен 0. Далее мы узнаем второй, третий и последний символ. В конце мы вышли в астрал, узнали правильный ответ и сообщили его программе жюри, на что получили положительный ответ. При решении задачи не пытайтесь выйти в астрал, доверьте это профессионалам.