Задача A. Отгадай число

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

Эта задача немного необычна — в ней вам предстоит реализовать интерактивное взаимодействие с тестирующей системой. Это означает, что вы можете делать запросы и получать ответы в online-режиме. Обратите внимание, что ввод/вывод в этой задаче — стандартный (то есть с экрана на экран). После вывода очередного запроса обязательно используйте функции очистки потока, чтобы часть вашего вывода не осталась в каком-нибудь буфере. Например, на C++ надо использовать функцию `fflush(stdout)`, на Java вызов `System.out.flush()`, на Pascal `flush(output)` и `stdout.flush()` для языка Python.

В этой задаче вам предстоит в интерактивном режиме угадать число x, которое загадала тестирующая система. Про загаданное число x известно, что оно целое и лежит в границах от 1 до n включительно (значение n известно заранее).

Вы можете делать запросы к тестирующей системе, каждый запрос — это вывод одного целого числа от 1 до n. Есть два варианта ответа тестирующей системы на запрос:

- строка «`<`» (без кавычек), если загаданное число меньше числа из запроса;
- строка «`>=`» (без кавычек), если загаданное число больше либо равно числу из запроса.

В случае, если ваша программа наверняка угадала нужное число x, выведите строку вида «`! x`», где x — это ответ, и завершите работу своей программы.

Вашей программе разрешается сделать не более 25 запросов.

Формат входных данных

Для чтения ответов на запросы программа должна использовать стандартный ввод.

В первой строке входных данных будет содержаться целое положительное число n ($1 \leqslant n \leqslant 10^6$) — максимально возможное число, которое может быть загадано.

В следующих строках на вход вашей программе будут подаваться строки вида «`<`» и «`>=`». i-я из этих строк является ответом системы на ваш i-й запрос. После того, как ваша программа угадала число, выведите «`! x`» (без кавычек), где x — это ответ, и завершите работу своей программы.

Тестирующая система даст вашей программе прочитать ответ на запрос из входных данных только после того, как ваша программа вывела соответствующий запрос системе и выполнила операцию `flush`.

Формат выходных данных

Для осуществления запросов программа должна использовать стандартный вывод.

Ваша программа должна выводить запросы — целые числа x_i ($1 \leqslant x_i \leqslant n$) по одному в строке (не забывайте выводить «*перевод строки*» после каждого значения x_i). После вывода каждой строки программа должна выполнить операцию `flush`.

Каждое из значений x_i обозначает очередной запрос к системе. Ответ на запрос программа сможет прочесть из стандартного ввода. В случае, если ваша программа угадала число x, выведите строку вида «`! x`» (без кавычек), где x — ответ, и завершите работу программы.

Примеры

стандартный ввод	стандартный вывод
20 5 < 3 >= 4 >= ! 4

Задача B. 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 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. Фишки

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

Это интерактивная задача. Ваша программа будет взаимодействовать с программой жюри, используя стандартный ввод и вывод.

Программа жюри решила сыграть с вашей программой в игру. На доске $n \times n$ в двух различных клетках находятся две фишки. Ваша программа должна определить положение фишек. Для этого она можете пытаться двигать фишки, а программа жюри будет сообщать результаты передвижений.

За один ход можно выбрать фишку и попросить переместить её на одну клетку влево, вправо, вверх или вниз. Программа жюри сообщает результат перемещения — если клетка в выбранном направлении существует и свободна, то перемещение считается успешным и фишка перемещается в эту клетку. В противном случае перемещение считается неудачным и фишка остается на той же клетке.

Вы выигрываете, если после очередного хода можете назвать исходное положение фишек на доске. Ваша задача — выиграть не более чем за $6n$ ходов.

Введем на доске систему координат таким образом, что клетки имеют координаты $(1, 1), (1, 2), \ldots, (1, n), (2, 1), \ldots, (n, n)$. Команды для перемещения фишки кодируется латинской буквой следующим образом:

- «`U`» — переместиться с клетки (x, y) на клетку $(x, y + 1)$.
- «`D`» — переместиться с клетки (x, y) на клетку $(x, y - 1)$.
- «`R`» — переместиться с клетки (x, y) на клетку $(x + 1, y)$.
- «`L`» — переместиться с клетки (x, y) на клетку $(x - 1, y)$.

Протокол взаимодействия

В самом начале программа жюри сообщает вашей программе натуральное число n ($2 \leqslant n \leqslant 50$) — размер доски.

Далее ваша программа должна повторять следующие ходы, выводя в стандартный поток вывода соответствующее сообщение и переводя строку.

- Если ваша программа считает, что определила начальное положение фишек, следует вывести 5 чисел: «1 x_1 y_1 x_2 y_2» ($1 \leqslant x_1, y_1, x_2, y_2 \leqslant n$) — начальное положение первой фишки (x_1, y_1) и второй фишки (x_2, y_2), соответственно. После вывода этой команды ваша программа должна завершиться.
- Если ваша программа хочет попытаться переместить фишку, следует вывести строку «0 id c», где id — номер фишки, которую ваша программа хочет переместить (1 или 2), а символ c — направление движения.

После каждого перемещения программа жюри сообщает вашей программе результат попытки перемещения:

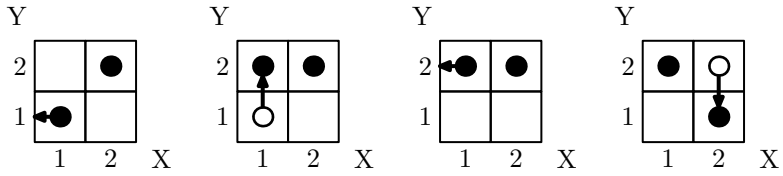
- «`1`», если передвижение успешное
- «`0`», если нет

Пример

стандартный ввод	стандартный вывод
2 0 1 0 1	0 1 L 0 1 U 0 1 L 0 2 D 1 1 1 2 2

Замечание

В примере фишки перемещались следующим образом.



В каждом тесте положение фишек исходно зафиксировано, и программа жюри честно отвечает на запросы вашей программы.

Задача E. Погоня в метро

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	0.25 секунд
Ограничение по памяти:	256 мегабайт

Это интерактивная задача. Параллельно с выполнением вашего решения жюри запускает проверяющую программу с который вы обмениваетесь сообщениями через стандартный ввод и вывод. Подробнее о протоколе взаимодействия написано ниже. Также в конце условия вы можете посмотреть корректные примеры взаимодействия с проверяющей программой на разных языках программирования.

В прекрасной Метрополии будущего необходимость в машинистах, управляющих поездами метро, отпала. Благодаря развитию технологий, их заменил искусственный интеллект (ИИ). К сожалению, в один прекрасный день опасения писателей-фантастов сбылись — ИИ взбунтовался, и теперь где-то в метро ездит неуправляемый поезд. Страшно представить, чем это грозит городской транспортной системе! Ваша задача состоит в том, чтобы найти поезд в сложной системе метро и остановить неуправляемый ИИ.

В целях безопасности все остальные поезда были отправлены в депо, а все ветки, кроме той, на которой находится неконтролируемый поезд, были перекрыты, поэтому на данный момент метро Метрополии представляет из себя одну ветку (обычную прямую ветку без самопересечений) из n станций, последовательно пронумерованных от 1 до n, ровно на одной из которых находится поезд. Для поимки неуправляемого поезда вам требуется определить номер этой станции, после чего на путях будут установлены искусственные заграждения и поезд будет пойман.

Для определения нужной станции диспетчер Сара одолжила вам устройство, позволяющее вам выбрать произвольные числа l и r ($l \leqslant r$), после чего оно проверит, верно ли, что поезд находится на станции с номером между l и r. К сожалению, для перезарядки устройства требуется k минут (и вы используете его как только перезарядка завершается), поэтому между двумя применениями поезд может перебраться из той станции, где он сейчас находится, в любую станцию с номером отличающимся не более чем на k. Формально, если при некотором применении устройства поезд находился на станции x, то при следующем применении он может находиться на любой станции y, такой что $max(1, x-k) \leqslant y \leqslant min(n, x+k)$. При этом поезд не знает, что вы пытаетесь его поймать и совершает все перемещения согласно некоторому заранее составленному им плану.

В процессе изучения устройства вы выяснили, что оно было сделано очень давно, и сможет выдержать не более чем q использований, после чего оно сломается, а ваша задача будет считаться проваленной.

Сможете ли вы найти станцию, на которой находится поезд, за не более чем q использований устройства?

Протокол взаимодействия

При запуске решения на вход подаются три целых числа n ($1 \leqslant n \leqslant 10^{18}$), k ($0 \leqslant k \leqslant 10$) и q ($4500 \leqslant q \leqslant 15\,000$).

Чтобы использовать устройство, вы должны вывести через пробел два числа l и r ($1 \leqslant l \leqslant r \leqslant n$). В ответ на это вы получите либо строку «`Yes`», если между станциями с номерами l и r находится поезд, либо строку «`No`» иначе. Если $l = r$ и вы получили ответ «`Yes`», это значит, что вы точно определили станцию, на которой находится поезд, и ваша программа после этого должна немедленно завершиться.

Если ваша программа за q запросов не смогла точно определить станцию, на которой находится поезд, программа должна также немедленно завершиться, в противном случае вердикт тестирующей системы может быть любым.

После каждого запроса необходимо вывести перевод строки и сбросить 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. В точности соблюдайте формат взаимодействия с системой.

Пример

стандартный ввод	стандартный вывод
10 2 15000 Yes No Yes	 3 4 3 3 2 2

Замечание

В первом тесте поезд изначально находился на станции 3, после первого использования устройства переместился на станцию 2, а после второго — остался на месте.

Задача F. Ехаб и еще одна очередная задача на xor

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

Это интерактивная задача!

Ехаб играет в игру с Лагги. Ехаб имеет два загаданных числа (a, b). Лагги может сказать пару чисел (c, d) и Ехаб ответит:

- 1 если $a \oplus c > b \oplus d$.
- 0 если $a \oplus c = b \oplus d$.
- -1 если $a \oplus c < b \oplus d$.

Операция $a \oplus b$ обозначает операцию побитовое исключающее «ИЛИ» чисел a и b.

Лагги нужно угадать (a, b) **не более, чем за 62 вопроса**. Вам предлагается сыграть в эту игру. Вы играете за Лагги, а программа жюри - за Ехаба.

Гарантируется, что $0 \leqslant a, b < 2^{30}$**.**

Формат входных данных

См. протокол взаимодействия.

Формат выходных данных

Чтобы вывести ответ, выведите "! a b"(без кавычек). **Не забудьте сбросить буфер вывода после того, как выведете ответ**.

Протокол взаимодействия

Чтобы задать вопрос, выведите "? c d"(без кавычек). c и d должны быть неотрицательными целыми числами, меньшими 2^{30}. **Не забудьте сбросить буфер вывода после того, как зададите вопрос**.

После каждого вопроса вы должны считать ответ. Если программа жюри отвечает числом -2, это значит, что ваша программа задала больше, чем 62 запроса и должна завершиться.

Чтобы сбросить буфер вывода вы можете использовать:

- fflush(stdout) в C++.
- System.out.flush() в Java.
- stdout.flush() в Python.
- flush(output) в Pascal.
- Прибегните к документации других языков.

Пример

стандартный ввод	стандартный вывод
1 -1 0	? 2 1 ? 1 2 ? 2 0 ! 3 1

Замечание

В примере из условия:

Загаданные числа: $a = 3$, $b = 1$.

В первом вопросе: $3 \oplus 2 = 1$ и $1 \oplus 1 = 0$, и ответ равен 1.

Во втором вопросе: $3 \oplus 1 = 2$ и $1 \oplus 2 = 3$, и ответ равен -1.

В третьем вопросе: $3 \oplus 2 = 1$ и $1 \oplus 0 = 1$, и ответ равен 0.

После этого программа выводит ответ и завершается.

Задача G. Странный прибор

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

Это интерактивная задача.

Вася обожает решать загадки и разгадывать головоломки. На этот раз он нашел странный прибор и хочет выяснить принцип его работы.

Этот прибор зашифрован с помощью дерева (связного неориентированного графа без циклов), состоящего из n вершин, пронумерованных целыми числами от 1 до n. Чтобы решить головоломку надо угадать это дерево.

К счастью прибор умеет выполнять одну операцию, исходя из которой надо разгадать его шифр. Можно ввести в прибор последовательность $d_1, d_2, \ldots, d_n$ целых неотрицательных чисел. На приборе есть n лампочек, i-я из которых отвечает за i-ю вершину дерева-шифра. Для всех i из этих лампочек i-я загорится, если существует такая вершина дерева-шифра с номером $j \neq i$, что $dist(i, j) \leqslant d_j$. Здесь $dist(i, j)$ обозначает расстояние между вершинами i и j в дереве-шифре, то есть количество ребер в простом пути между вершинами i и j.

Вася хочет за $\leqslant 80$ операций с прибором решить головоломку и угадать дерево-шифр. Помогите ему!

Протокол взаимодействия

В начале вашей программе вводится единственное целое число n — количество вершин в дереве-шифре прибора ($2 \leqslant n \leqslant 1000$).

Далее вы можете выполнять операции в следующем формате. Сначала выведите символ "`?`" (без кавычек) и за ним n целых чисел $d_1, d_2, \ldots, d_n$, разделенных пробелами. Заметьте, что в операциях для всех i должно быть выполнено неравенство $0 \leqslant d_i < n$. В ответ будет выведена строка s длины n, состоящая из символов "`0`" и "`1`" (без кавычек). Для всех i символ s_i будет равен "`0`", если у прибора не включилась лампочка, соответствующая i-й вершине дерева-шифра и "`1`", иначе.

После нескольких запросов вы должны вывести угаданное дерево. Для этого в первой строке выведите единственный символ "`!`" (без кавычек). В следующих $n - 1$ строках выведите по 2 целых числа a_i, b_i — номера вершин, соединяющих i-е ребро дерева. Выведенные числа должны удовлетворять условиям $1 \leqslant a_i, b_i \leqslant n$ и $a_i \neq b_i$. Эти ребра должны образовывать дерево, совпадающее с загаданным. Выводить ребра можно в любом порядке. После этого ваша программа должна завершиться.

Гарантируется, что в каждом тесте дерево-шифр будет зафиксировано заранее и не будет меняться в зависимости от выполняемых операций.

Ваша программа может выполнить от 0 до 80 операций с прибором и после этого сообщить дерево, совпадающее с загаданным.

Если ваша программа сделает больше 80 операций, то она может получить любой вердикт, потому что будет считывать данные из закрытого потока ввода. Также, если ваша программа сделает операцию или выведет ответ в неверном формате, она может получить любой вердикт. **Будьте внимательны.**

Не забудьте сбрасывать буфер вывода после того, как выведете данные для операции или ответ.

Чтобы сбросить буфер вывода вы можете использовать:

- fflush(stdout) в C++.
- System.out.flush() в Java.
- stdout.flush() в Python.
- flush(output) в Pascal.
- Прибегните к документации других языков.

Пример

стандартный ввод	стандартный вывод
5 00000 11011 11100 10010	? 0 0 0 0 0 ? 1 1 2 0 2 ? 0 0 0 1 0 ? 0 1 0 0 1 ! 4 2 1 5 3 4 4 1

Замечание

Таблица попарных расстояний между вершинами выглядит так:

	1	2	3	4	5
1	0	2	2	1	1
2	2	0	2	1	3
3	2	2	0	1	3
4	1	1	1	0	2
5	1	3	3	2	0

- Если сделать операцию, в которой $d = [0, 0, 0, 0, 0]$, то ни одна лампочка не загорится, потому что $dist(i, j) > 0$ при всех $i \neq j$.
- Если сделать операцию, в которой $d = [1, 1, 2, 0, 2]$, то загорятся все лампочки, кроме лампочки, соответсвующей вершине дерева-шифра с номером 3. Например, лампочка, соответсвующая вершине с номером 1 загорится, потому что $dist(1, 5) = 1 \leqslant 2 = d_5$.
- Если сделать операцию, в которой $d = [0, 0, 0, 1, 0]$, то загорятся все лампочки, кроме лампочек, соответсвующих вершинам дерева-шифра с номерами 4 и 5.
- Если сделать операцию, в которой $d = [0, 1, 0, 0, 1]$, то загорятся только лампочки, соответсвующие вершинам дерева-шифра с номерами 1 и 4.

Задача H. Лошадь и Ежик

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

Это интерактивная задача.

Ёжик спустился в туман и оказался в прямоугольной долине размером N на M метров, по которой бродит Лошадь. Ёжик хочет ее найти. Будем считать, что в каждый момент времени и Ёжик, и Лошадь находятся в одной из $N \times M$ клеток. Туман настолько густой, что Лошадь не видно, даже если она находится в той же самой клетке , что и Ёжик. К счастью Ёжик обладает очень острым слухом и может понять, в каком направлении сместилась Лошадь. Он также может позвать Лошадь, и, если она находится в одной клетке с Ёжиком, то Лошадь его услышит и обязательно отзовется.

В каждый момент времени Ёжик может сместиться в соседнюю клетку по горизонтали, вертикали или диагонали. Потом он отчетливо слышит, куда сместилась Лошадь относительно своего старого местоположения. Лошадь за единицу времени смещается на одну клетку только по горизонтали или вертикали (влево, вверх, вправо или вниз). При этом Лошадь не выходит за границы долины, поэтому и Ёжик не должен этого делать.

Требуется написать программу, которая поможет Ёжику, знающему свое начальное положение и следящему за передвижениями Лошади, как можно быстрее ее найти. Кроме того, количество запросов о том, находится ли Лошадь в одной клетке с Ёжиком, не должно превышать $N \times M$.

Протокол взаимодействия

Сначала программа-решение должна прочитать из стандартного потока ввода натуральные числа N и M, записанные в первой строке, а из второй строки координаты начального местоположения Ёжика — два натуральных числа: x_0 — номер столбца, y_0 — номер строки ($1 \leqslant x_0 \leqslant M$, $1 \leqslant y_0 \leqslant N$). Числа в каждой строке разделены пробелом. Затем программа-решение начинает взаимодействие с программой, моделирующей поведение лошади, в соответствии со следующим протоколом:

1. Программа выводит в стандартный поток вывода одну строку, описывающую ход Ёжика, которая содержит три числа: его перемещение в виде указания смещения по горизонтали dx ($dx = -1,\ 0$ или 1) и по вертикали dy ($dy = -1,\ 0$ или 1), а также число 1, если Ёжик зовет Лошадь в клетке, в которую он при этом попадет, или 0 — если не зовет.

2. После этого программа должна считать из стандартного потока ввода ответ программы, сообщающей о действии Лошади. Ответ состоит из трех чисел, расположенных в одной строке через пробел. Первое число ответа может быть равно 0 или 1, где

   - 0 означает, что Ёжик не пытался позвать Лошадь либо позвал, но Лошади в его клетке нет. В этом случае следующие два числа обозначают очередное смещение Лошади по горизонтали dx ($dx = -1,\ 0$ или 1) и по вертикали dy ($dy = -1,\ 0$ или 1), при этом хотя бы одно из значений dx или dy равно нулю;
   - 1 означает, что Ёжик позвал Лошадь, и она действительно оказалась в той же клетке, что и он. В этом случае другие два числа равны 0, и программа-решение должна закончить свою работу.

Ваша программа должна сделать не более 10 000 ходов. Положительный вердикт на тесте вы получите, если выполнено следующее условие:

$$10 \cdot (\frac{J}{S})^2 \geqslant 9.5$$

Где J — это общее число ходов в программе жюри, а S — вашей программы. В данной задаче чекер адаптивный, и пытается играть с вами как можно дольше.

Пример

стандартный ввод	стандартный вывод
2 3 1 2 0 1 0 0 0 0 1 0 0	0 0 1 1 -1 0 1 0 1

Замечание

Ёжик находился в клетке (1, 2). Сначала он попробовал позвать Лошадь в той же клетке (вывод: 0 0 1), но Лошади там не оказалось, и она сместилась вправо (ввод: 0 1 0). Ёжик сместился по диагонали, но Лошадь звать не стал (вывод: 1 –1 0), а Лошадь осталась на месте (ввод: 0 0 0). Ежик сместился вправо и позвал Лошадь (вывод: 1 0 1). Лошадь оказалась в той же клетке и отозвалась (ввод: 1 0 0). Значит, изначально Лошадь находилась в клетке (2, 1), а встретились они в клетке (3, 1). Ёжик при этом сделал три хода и дважды запросил местоположение Лошади.

Соблюдайте формат ввода-вывода, очищайте за собой буфер, и не теряйте друзей.