Задача A. Детская игра с роботом

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Это интерактивная задача.

Маленький Миша играет с роботом в детскую игру. В этой игре робот находится на клетчатом поле 3×3, огороженном барьерами. Игра начинается в центральной клетке и состоит в выполнении десяти действий. Каждое действие — либо декламация вслух какой-то фразы, либо попытка переместиться в одну из соседних клеток. Декламация используется, если нужно подождать, или же просто для развлечения.

Одна из клеток поля — особенная, но Миша не знает заранее, какая это клетка. При перемещении в особенную клетку робот сообщает об этом. Цель игры — переместиться в особенную клетку ровно на десятом действии.

Для общения с роботом используется текстовый ввод и вывод. Поддерживаются следующие команды:

- `echo` *phrase* — декламация фразы *phrase*,
- `move north` — перемещение на одну клетку на север,
- `move east` — перемещение на одну клетку на восток,
- `move south` — перемещение на одну клетку на юг,
- `move west` — перемещение на одну клетку на запад.

В ответ на команду декламации робот декламирует заданную фразу, а также выводит её в текстовом виде. При декламации робот никуда не перемещается. Гарантируется корректная работа робота, если фраза состоит из символов с ASCII-кодами от 32 до 126, включительно, а её длина не превосходит 256 символов.

В ответ на команду перемещения робот выводит одно из четырёх слов:

- `bump`, если вместо перемещения робот упёрся в барьер, в этом случае робот остаётся на месте,
- `moved`, если робот успешно переместился в обычную клетку,
- `found`, если робот переместился в особенную клетку на одном из первых девяти действий,
- `win`, если робот переместился в особенную клетку на десятом действии.

Помогите Мише сыграть в эту игру так, чтобы на десятом действии получить в ответ от робота заветное слово `win`.

Протокол взаимодействия с программой жюри:

Решение должно выводить каждую команду в стандартный поток вывода на отдельной строке. Чтобы предотвратить буферизацию вывода, после каждой выведенной строки следует вставить команду очистки буфера вывода: например, это может быть `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

Ответ на команду решение получает в стандартный поток ввода, также на отдельной строке.

После вывода десяти команд решение должно корректно завершить свою работу.

Пример

В приведённом примере особенная клетка лежит на востоке от центральной. Гарантируется, что этот тест будет первым при проверке.

Слева приведены команды (**вывод** решения участника), а справа — ответы на них (**ввод** для решения участника).

команда	ответ
`moved`	`move north`
`moved`	`move east`
`found`	`move south`
`moved`	`move west`
`found`	`move east`
`bump`	`move east`
`moved`	`move south`
`Ready!`	`echo Ready!`
`Steady...`	`echo Steady...`
`win`	`move north`

Комментарий

В каждом тесте особенная клетка выбрана заранее и не меняет своё положение при повторной проверке.

Задача B. Поиск маленьких чисел

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Это — интерактивная задача.

Жюри загадало перестановку чисел от 1 до n. Ваша задача — найти позиции, в которых стоят числа от 1 до k. Для этого вы можете воспользоваться программой жюри, которая умеет сравнивать числа, стоящие на двух позициях в перестановке.

Формат входного файла

В первой строке будет задано два числа n и k — размер перестановки и количество чисел, позиции которых надо найти. Во всех тестах, кроме теста из примера, $n = 10\,000$, а $k \le 10$.

Далее будут следовать ответы на ваши запросы по одному в строке. Если первое число из сравниваемых меньше, то в строке будет содержаться единственный символ «<», иначе — единственный символ «>».

Формат выходного файла

Если вы хотите сравнить числа на i-й и j-й позициях, необходимо вывести строку «? i j». При этом i и j должны быть различными целыми числами от 1 до n. Вы можете сделать не более 10 700 таких запросов.

Если вы нашли позиции всех чисел от 1 до k, то необходимо вывести «! pos_1 pos_2 ... pos_k», после чего завершить работу программы.

Чтобы предотвратить буферизацию вывода, после каждой выведенной строки следует вставить команду очистки буфера вывода: например, это может быть `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

Также не забывайте выводить символ перевода строки в конце каждой строки, которую вы выводите.

Пример

стандартный ввод	стандартный вывод
3 3 *(waiting for output)* < *(waiting for output)* > *(waiting for output)* <	*(reading input)* ? 1 2 *(reading input)* ? 3 1 *(reading input)* ? 2 3 *(reading input)* ! 1 2 3 *(terminating)*

Комментарий

В примере загадана перестановка 1 2 3.

Задача C. Случайные тоннели

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Это интерактивная задача.

В галактике Туманный Путь бесконечное количество звёзд, пронумерованных целыми числами. Чтобы быстро перемещаться между ними, можно пользоваться червоточинами — надпространственными тоннелями, соединяющими звёзды. Однако не любая пара звёзд соединена таким тоннелем.

Когда демиург создавал Туманный Путь, он задал вероятность p того, что для любой пары целых чисел (i, j) от звезды i есть прямой тоннель к звезде j, а далее предоставил конструирование тоннелей воле случая. Вероятность существования каждого тоннеля не зависит от вероятности существования любых других тоннелей. Все тоннели односторонние, то есть при $i \neq j$ тоннель от i к j и тоннель от j к i — это два разных тоннеля, каждый из которых существует с вероятностью p.

Звездолёт находится у звезды с номером 0. Навигационный компьютер звездолёта имеет ограниченную функциональность: он может принимать запрос вида «следует переместиться по прямому тоннелю к звезде x», где x — целое число от 0 до $2^{32} - 1$. После такого запроса, если существует тоннель от звезды, где находится звездолёт, к звезде x, звездолёт перемещается туда, а на табло загорается слово «`yes`». В противном случае на табло появляется слово «`no`», а звездолёт остаётся на месте.

Капитан звездолёта хочет попасть к звезде с номером 1. Вы — навигатор этого звездолёта. Помогите капитану оказаться у нужной звезды!

Протокол взаимодействия с программой жюри:

Решение должно выводить каждый запрос в стандартный поток вывода на отдельной строке. Запрос — это одно целое число x от 0 до $2^{32} - 1$ — номер звезды, к которой следует переместить звездолёт при наличии прямого тоннеля.

Чтобы предотвратить буферизацию вывода, после каждого выведенного запроса следует вставить команду очистки буфера вывода: например, это может быть `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

Ответ на запрос — слово «`yes`» или «`no`» — решение получает в стандартный поток ввода, также на отдельной строке.

Как только звездолёт оказался у звезды с номером 1, решение должно сразу же корректно завершить свою работу.

После 30 000 запросов к навигационному компьютеру у него кончается электричество, и следующий запрос сделать не удаётся. В таком случае миссия считается проваленной.

В каждом тесте к этой задаче зафиксирована вероятность p (целое число от 3 до 99 в процентах). После этого для каждой возможной пары звёзд зафиксировано, есть ли между ними тоннель.

Пример

запросы участника	ответы проверяющей программы
1	no
3	yes
3	no
1	yes

Комментарий

Обратите внимание: **слева** указан **вывод** программы участника, а **справа** — то, что она после этого получает **на вход**.

В примере рассматривается первый тест в системе. Вероятность существования каждого тоннеля в нём равна 50%. Прямого тоннеля от звезды 0 к звезде 1 нет. Зато удаётся переместиться от звезды 0 к звезде 3. После этого навигатор хочет переместиться от звезды 3 к самой себе, но такого туннеля нет. Наконец, туннель от звезды 3 к звезде 1 существует, и миссию удаётся выполнить.

Задача D. Вариация крестиков-ноликов

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Это интерактивная задача.

Дима играет с Петей в вариант игры «крестики-нолики». Для игры используется поле из 3×3 клеток, исходно каждая клетка пуста. Игроки делают ходы по очереди, первым ходит Дима. Каждый игрок в свой ход должен поставить свой символ (Дима ставит крестики, а Петя — нолики) в любую пустую клетку.

Дима выигрывает, если поставит три крестика в ряд: по горизонтали, вертикали или одной из двух диагоналей. Правила для ноликов отличаются от обычных: задача Пети — **не дать Диме построить ряд** из трёх крестиков. Формально Петя выигрывает, если во всех клетках доски уже стоят какие-то символы, но Дима ещё не выиграл.

Ваша задача — играть за Диму так, чтобы всегда выигрывать. Программа жюри будет играть за Петю.

Протокол взаимодействия с программой жюри:

В каждом тесте вашей программе нужно сыграть за Диму от 1 до 100 партий в описанную игру. Тесты могут отличаться количеством партий и стратегией Пети. Каждая партия состоит из нескольких ходов.

В течение партии играющие программы обмениваются текущим положением на доске. Положение задаётся тремя строками, каждая из которых содержит по три символа. Символ «`x`» (английская маленькая буква икс) означает, что в соответствущей клетке стоит крестик, символ «`o`» (английская маленькая буква о) — нолик, а символ «`.`» (точка) говорит о том, что клетка пуста.

Каждая партия начинается с того, что программе участника на вход подаётся начальное положение на доске: все клетки пусты. Чтобы сделать ход, каждая из играющих программ должна вывести только что полученную позицию, изменённую в соответствии с правилами хода: ровно одна из пустых клеток должна стать клеткой с символом соответствующего игрока.

Партия заканчивается, когда один из игроков выигрывает. При этом, если выиграл Петя, проверка завершается с вердиктом «`Wrong Answer`». Если же выиграл Дима, игроки начинают следующую партию, если она запланирована.

В каждом тесте заранее определено, сколько партий будет сыграно и как будет играть Петя в каждой из них. Если все запланированные партии закончились победой Димы, то вместо пустой позиции, с которой начинается партия, программа участника получает на вход позицию, в которой все клетки **заняты крестиками**. Получив на вход такую позицию, программа участника должна корректно завершить свою работу.

Чтобы предотвратить буферизацию вывода, после каждой выведенной позиции следует вставить команду очистки буфера вывода: например, это может быть `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

Пример

стандартный ввод	стандартный вывод
...	*(reading input)*
...	*(reading input)*
...	*(reading input)*
(waiting for output)	...
(waiting for output)	.x.
(waiting for output)	...
o..	*(reading input)*
.x.	*(reading input)*
...	*(reading input)*
(waiting for output)	ox.
(waiting for output)	.x.
(waiting for output)	...
oxo	*(reading input)*
.x.	*(reading input)*
...	*(reading input)*
(waiting for output)	oxo
(waiting for output)	.x.
(waiting for output)	.x.
...	*(reading input)*
...	*(reading input)*
...	*(reading input)*
(waiting for output)	...
(waiting for output)	.x.
(waiting for output)	...
o..	*(reading input)*
.x.	*(reading input)*
...	*(reading input)*
(waiting for output)	o..
(waiting for output)	.x.
(waiting for output)	.x.
oo.	*(reading input)*
.x.	*(reading input)*
.x.	*(reading input)*
(waiting for output)	oo.
(waiting for output)	.x.
(waiting for output)	xx.
ooo	*(reading input)*
.x.	*(reading input)*
xx.	*(reading input)*
(waiting for output)	ooo
(waiting for output)	.x.
(waiting for output)	xxx
xxx	*(reading input)*
xxx	*(reading input)*
xxx	*(reading input)*
	(terminating)

Комментарий

В примере, который также является первым тестом при проверке, необходимо сыграть две партии. В каждой партии Петя на каждом ходу ставит нолик в самую верхнюю пустую клетку. Если таких клеток несколько, он выбирает самую левую из них.

Обратите внимание: пустых строк во вводе и выводе на самом деле **нет**. Пропуски добавлены для того, чтобы все строки ввода и вывода шли в порядке времени, в которое они были переданы.

Задача E. Just Another Disney Problem

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дано N, жюри загадывает турнир, можно задавать запрос, в какую сторону направлено ребро из X в Y. Нужно вывести топологически непротиворечивую перестановку всех вершин (то есть, если вершина A идет в списке раньше B, то есть путь из A в B).

Формат входного файла

Первое число — N ($1 \le N \le 1000$). На каждый запрос ответ — строка "`YES`", если дуга идёт из X в Y, или "`NO`", если дуга идёт из Y в X.

Формат выходного файла

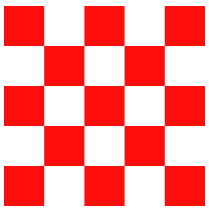
Запросы участника — три числа 1, X, Y ($1 \le X, Y \le N$, $X \ne Y$). Ограничение на количество запросов — 9000. В последней строке ответ: строка "`0 `", далее N чисел a_i ($1 \le a_i \le N$) через пробел — искомая перестановка.

Примеры

stdin	stdout
2 NO	1 1 2 0 2 1

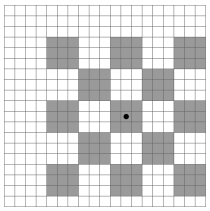
Задача F. Инопланетяне

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт



Мирко — большой фанат выкошенных на полях и лугах кругов и других геометрических объектов предположительно инопланетного происхождения. Одной летней ночью он решил выкосить свой собственный геометрический объект на лугу своей бабушки. Как большой патриот (а также как пациент клиники для душевнобольных), Мирко решил выкосить объект, который будет иметь форму щитовой части хорватского герба, которая представляет собой шахматную доску размером 5×5 из 13 красных квадратов и 12 белых квадратов.

Бабушкин луг представляет собой квадрат, разделённый на $N \times N$ квадратных ячеек одинакового размера. Ячейка в левом нижнем углу имеет координаты $(1, 1)$, а ячейка в правом верхнем углу — координаты (N, N). Мирко решил скосить траву только на тех ячейках, которые образуют на гербе квадраты красного цвета, и оставить остальную траву нескошенной. Он выбрал нечётное целое число $M \geq 3$ и скосил траву таким образом, что каждый квадрат шахматной доски соответствует $M \times M$ ячейкам луга, и шахматная доска целиком располагается внутри луга.



Пример луга и выкошенного Мирко объекта. Здесь $N = 19$ и $M = 3$. Ячейки, где трава скошена, показаны серым цветом. Центр объекта находится в ячейке с координатами $(12, 9)$, отмеченной чёрной точкой.

После того, как Мирко пошел спать, его странное творение привлекло внимание настоящих инопланетян! Летая над лугом в своём космическом корабле, они исследуют с помощью простого инопланетного устройства объект, выкошенный Мирко. Это устройство может только определять, скошена трава в некоторой ячейке или нет. Инопланетяне обнаружили одну ячейку со скошенной травой и теперь хотят найти центральную ячейку творения Мирко, чтобы восхититься его красотой. Однако, они не знают размер квадрата M в объекте, выкошенном Мирко.

Напишите программу, которая по заданному размеру луга N ($15 \leq N \leq 2\,000\,000\,000$) и координатам (X_0, Y_0) одной из ячеек со скошенной травой находит координаты центральной ячейки выкошенного Мирко объекта посредством общения с инопланетным устройством. На каждом тесте устройство можно использовать не более 300 раз.

Формат входного файла

Это интерактивная задача. Ваша программа посылает команды инопланетному устройству с использованием стандартного потока вывода и получает ответы из стандартного потока ввода. Каждая строка с командой должна завершаться одним переводом строки.

- В начале работы вашей программы вы должны прочитать три целых числа N, X_0 и Y_0, разделённые одиночными пробелами, из стандартного потока ввода. Число N — это размер луга, а (X_0, Y_0) — это координаты одной ячейки со скошенной травой.
- Чтобы проверить с использованием инопланетного устройства, скошена ли трава в ячейке (X, Y), необходимо вывести в стандартный поток вывода строку в формате `examine X Y`. Если координаты (X, Y) не находятся внутри луга (не выполены условия $1 \le X \le N$ и $1 \le Y \le N$), или вы используете устройство более 300 раз, ваша программа получает 0 баллов на этом тесте.
- Инопланетное устройство будет отвечать одной строкой `true`, если трава в ячейке (X, Y) скошена, в противном случае оно будет отвечать строкой `false`.
- Когда ваша программа нашла центральную ячейку, она должна вывести строку вида `solution` X_C Y_C в стандартный поток вывода, где (X_C, Y_C) — координаты центральной ячейки. Исполнение вашей программы будет автоматически завершено, как только она выведет решение.

Для того, чтобы правильно общаться с устройством, ваша программа должна делать операцию `flush`. Для очистки буфера необходимо выполнить команду `flush(output)` в `Pascal`, `fflush(stdout)` в `C`, `cout.flush()` в `C++`.

Примеры

стандартный ввод	стандартный вывод
19 7 4 true false false true	 examine 11 2 examine 2 5 examine 9 14 examine 18 3 solution 12 9

Комментарий

Пустые строки приведены для наглядности порядка ввода и вывода. Каждая пустая строка вывода после команды соответствует операции `flush`.

Задача G. Jump

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

Consider a toy interactive problem OneMax which is defined as follows. You know an integer n and there is a hidden bit string S of length n. The only thing you may do is to present the system a bit string Q of length n, and the system will return the number OneMax(Q) — the number of bits which coincide in Q and S at the corresponding positions. The name of OneMax problem stems from the fact that this problem is simpler to explain when $S = 111\ldots11$, so that the problem turns into maximization (Max) of the number of ones (One).

When n is even, there is a similar (but harder) interactive problem called Jump. The simplest way to describe the Jump is by using OneMax:

$$\textsc{Jump}(Q) = \begin{cases} \textsc{OneMax}(Q) & \text{if } \textsc{OneMax}(Q) = n \text{ or } \textsc{OneMax}(Q) = n/2; \\ 0 & \text{otherwise.} \end{cases}$$

Basically, the only nonzero values of OneMax which you can see with Jump are n (which means you've found the hidden string S) and $n/2$.

Given an even integer n — the problem size, you have to solve the Jump problem for the hidden string S by making interactive Jump queries. Your task is to eventually make a query Q such that $Q = S$.

Протокол взаимодействия с программой жюри:

First, the testing system tells the length of the bit string n. Then, your solution asks the queries and the system answers them as given by the Jump definition. When a solution asks the query Q such that $Q = S$, the system answers n and terminates, so if your solution, after reading the answer n, tries reading or writing anything, it will fail.

The limit on the number of queries is $n + 500$. If your solution asks a $(n + 501)$-th query, then you will receive the "Wrong Answer" outcome. You will also receive this outcome if your solution terminates too early.

If your query contains wrong characters (neither `0`, nor `1`), or has a wrong length (not equal to n), the system will terminate the testing and you will receive the "Presentation Error" outcome.

You will receive the "Time Limit Exceeded" outcome and other errors for the usual violations.

Finally, if everything is OK (e.g. your solution finds the hidden string) on every test, you will receive the "Accepted" outcome, in this case you will have solved the problem.

Формат входного файла

The first line of the input stream contains an even number n ($2 \le n \le 1000$). The next lines of the input stream consist of the answers to the corresponding queries. Each answer is an integer — either 0, $n/2$, or n. Each answer is on its own line.

Формат выходного файла

To make a query, print a line which contains a string of length n which consists of characters `0` and `1` only. Don't forget to put a newline character and to flush the output stream after you print your query.

Пример

стандартный ввод	стандартный вывод
2 1 0 1 2	01 11 10 00

Задача D. Jump

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

Рассмотрим сначала простую интерактивную задачу, которую называют OneMax. Она определяется следующим образом. Вы знаете число n и загадана неизвестная строка S из нулей и единиц длины n. Вы можете делать запрос в виде строки Q из n нулей и единиц и вам отвечают число OneMax(Q) — количество позиций, в которых строки S и Q совпадают.

Пусть теперь n четно. Рассмотрим следующее усложнение задачи OneMax, которая называется Jump. Аналогично загадана строка S длины n из нулей и единиц, и можно делать запросы в виде строки Q из n нулей и единиц, при этом возвращаемое значение определяется по следующей формуле:

$$\textsc{Jump}(Q) = \begin{cases} \textsc{OneMax}(Q) & \text{if } \textsc{OneMax}(Q) = n \text{ or } \textsc{OneMax}(Q) = n/2; \\ 0 & \text{otherwise.} \end{cases}$$

Ваша задача решить задачу Jump, угадав строку.

Протокол взаимодействия с программой жюри

Сначала программа жюри присылает вам число n (n четно, $2 \le n \le 1000$). Затем вы можете задавать вопросы в виде битовых строк длины n, на каждый запрос Q программа жюри присылает значение Jump(Q). Если ответ на запрос равен n, то ваша программа должна завершиться, тест пройден.

Всего разрешается сделать $n + 500$ запросов.

Пример

стандартный ввод	стандартный вывод
2 1 0 1 2	01 11 10 00