Задача A. Поиск целого числа

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

Жюри загадывает целое число x от 1 до 10^9. Участник задаёт до 30 вопросов вида «это число y?». На каждый вопрос жюри отвечает либо «нет, $y < x$», либо «да, $y = x$», либо «нет, $y > x$». Задача участника — получить ответ «да».

Интерактор не адаптивный: жюри загадывает число заранее и честно отвечает на все вопросы.

Протокол взаимодействия

Чтобы задать вопрос «это число y?» (y целое, $1 \leqslant y \leqslant 10^9$), выведите на отдельной строке число y. Не забудьте вывести перевод строки и очистить буфер вывода, чтобы не получить вердикт `IL` (`Idleness Limit Exceeded`).

После этого прочитайте строку с одним символом:

- «<», если $y < x$,
- «=», если $y = x$,
- «>», если $y > x$.

Если вы добились ответа «=» — сразу завершите работу программы, чтобы получить вердикт `OK` (`Accepted`).

Если вы задали 30 вопросов, но так не добились ответа «=» — сразу завершите работу программы, чтобы точно получить вердикт `WA` (`Wrong Answer`), а не какой-то ещё.

Пример

стандартный ввод	*стандартный вывод*
< > =	4 6 5

Задача B. Поиск в сортированном массиве

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

Жюри загадывает массив длины $n \leqslant 1000$, строго отсортированный по возрастанию и состоящий из целых чисел от 1 до 10^9. Жюри также выбирает целое число x — один из элементов массива — и сообщает его участнику. Участник задаёт до 10 вопросов вида «какой элемент на позиции i?». На каждый вопрос жюри раскрывает указанный элемент. Задача участника — получить ответ x.

Интерактор не адаптивный: жюри загадывает массив заранее и честно отвечает на все вопросы.

Протокол взаимодействия

В первой строке заданы два целых числа n и x — размер массива и один из его элементов ($1 \leqslant n \leqslant 1000$; $1 \leqslant x \leqslant 10^9$).

Чтобы узнать элемент на позиции i ($1 \leqslant i \leqslant n$), выведите строку вида «`?` i». Не забудьте вывести перевод строки и очистить буфер вывода, чтобы не получить вердикт `IL` (`Idleness Limit Exceeded`).

После этого прочитайте строку, содержащую одно целое число y — элемент массива на позиции i.

Если вы добились ответа x — сразу завершите работу программы, чтобы получить вердикт `OK` (`Accepted`).

Если вы задали 10 вопросов, но так не добились ответа x — сразу завершите работу программы, чтобы точно получить вердикт `WA` (`Wrong Answer`), а не какой-то ещё.

Пример

стандартный ввод	*стандартный вывод*
3 6 2 9 6	 ? 1 ? 3 ? 2

Задача C. Поиск суммы двух

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

Жюри загадывает массив длины $n \leqslant 1000$, строго отсортированный по возрастанию и состоящий из целых чисел от 1 до 10^9. Жюри также выбирает два элемента массива — возможно, один и тот же два раза — и сообщает участнику их сумму s. Задача участника — найти два элемента с такой суммой.

Участник задаёт до 1000 вопросов вида «верно ли, что сумма элементов на позициях i и j равна s?». На каждый вопрос жюри отвечает либо «нет, их сумма меньше s», либо «да, их сумма равна s», либо «нет, их сумма больше s». Задача участника — получить ответ «да».

Интерактор не адаптивный: жюри загадывает массив заранее и честно отвечает на все вопросы.

Протокол взаимодействия

В первой строке заданы два целых числа n и s — размер массива и сумма двух его элементов ($1 \leqslant n \leqslant 1000$; $2 \leqslant s \leqslant 2 \cdot 10^9$).

Чтобы задать вопрос «верно ли, что сумма элементов на позициях i и j равна s?» ($1 \leqslant i, j \leqslant n$), выведите строку вида «`? i j`». Не забудьте вывести перевод строки и очистить буфер вывода, чтобы не получить вердикт `IL` (`Idleness Limit Exceeded`).

После этого прочитайте строку с одним символом:

- «`<`», если их сумма меньше s,
- «`=`», если их сумма равна s,
- «`>`», если их сумма больше s.

Если вы добились ответа «`=`» — сразу завершите работу программы, чтобы получить вердикт `OK` (`Accepted`).

Если вы задали 1000 вопросов, но так не добились ответа «`=`» — сразу завершите работу программы, чтобы точно получить вердикт `WA` (`Wrong Answer`), а не какой-то ещё.

Пример

В примере загадан массив $(2, 6, 9)$.

стандартный ввод	*стандартный вывод*

`3 11`
 	
`<`
 	
`>`
 	
`=`
	
	`? 1 2`
	
	`? 2 3`
	
	`? 3 1`

Задача D. Поиск максимума

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

Жюри загадывает массив длины $n \leqslant 1000$, состоящий из целых чисел от 1 до 10^9. Массив выпуклый: для $1 < i < n$ верно $a_{i-1} + a_{i+1} < 2 \cdot a_i$. Задача участника — найти максимальный элемент этого массива.

Участник задаёт до 20 вопросов вида «какой элемент на позиции i?». На каждый вопрос жюри раскрывает указанный элемент. После этого участник должен вывести значение максимального элемента в массиве.

Интерактор не адаптивный: жюри загадывает массив заранее и честно отвечает на все вопросы.

Протокол взаимодействия

В первой строке задано целое число n — размер массива ($1 \leqslant n \leqslant 1000$).

Чтобы задать вопрос «какой элемент на позиции i?» ($1 \leqslant i \leqslant n$), выведите строку вида «`?` i». Не забудьте вывести перевод строки и очистить буфер вывода, чтобы не получить вердикт `IL` (`Idleness Limit Exceeded`). Можно задать не более 20 таких вопросов.

После этого прочитайте строку, содержащую одно целое число — элемент массива на позиции i.

Чтобы дать ответ, выведите строку вида «`!` m» ($1 \leqslant m \leqslant 10^9$) и завершите работу программы.

Пример

В примере загадан массив $(2, 9, 6)$.

стандартный ввод	*стандартный вывод*
3 2 9 6	? 1 ? 2 ? 3 ! 9

Задача E. Джек и Джилл

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

Джек и Джилл играют в «угадай число». Сначала Джилл должна загадать целое число от 1 до 10^9. После этого Джек задаёт вопросы вида «верно ли, что это число x?», с каким-то целым x от 1 до 10^9. На каждый вопрос Джилл должна ответить либо «да», либо «нет, моё число больше», либо «нет, моё число меньше». Игра заканчивается, когда Джек угадывает число, или после 100 вопросов, если это так и не произошло.

Как Джек ни старается, ему никогда не удаётся угадать число меньше чем за 30 вопросов. Он понял, что Джилл жульничает: не загадывает число в самом начале, а отвечает на вопросы так, чтобы игра продолжалась достаточно долго. Джек задумался: как же она это делает?

Это интерактивная задача: вы играете за Джилл, а жюри — за Джека. Ваша задача — отвечать на вопросы так, чтобы Джек задал хотя бы 30 вопросов, прежде чем игра закончится. Имейте в виду, что ваши ответы не должны противоречить друг другу, иначе Джек сразу раскроет обман!

Протокол взаимодействия

Игра состоит из ходов. Каждый ход начинается с того, что жюри на отдельной строке сообщает целое число x ($1 \leqslant x \leqslant 10^9$) — то число, про которое Джек спросил «верно ли, что загаданное число равно x?». В ответ решение должно вывести на отдельной строке один символ: «=», если Джилл отвечает «да», или «>», если Джилл отвечает «нет, моё число больше», или «<», если Джилл отвечает «нет, моё число меньше».

После вывода строки с символом следует очистить буфер вывода: это можно сделать, например, вызовом `fflush(stdout)` в C или C++, `System.out.flush()` в Java, `flush(output)` в Pascal или `sys.stdout.flush()` в Python.

Если ответы Джилл противоречивы, или Джилл отвечает «да», или сделано уже 100 ходов, игра сразу заканчивается, и решение должно корректно завершить работу. Решение проходит тест, если ответы были непротиворечивы, а ходов было сделано не менее 30.

В различных тестах Джек использует различные стратегии для игры.

Примеры

стандартный ввод	*стандартный вывод*
1 2 *...и так далее...* 29 30	> > *...и так далее...* > =
1000 999 *...и так далее...* 902 901	< < *...и так далее...* < <

Пояснения к примерам

В первом примере Джек говорит числа 1, 2, 3, и так далее. Во втором примере Джек говорит числа 1000, 999, 998, и так далее. Гарантируется, что в первых двух тестах он будет следовать этим двум стратегиям.

В обоих примерах Джилл решила просто заранее загадать число 30. Ваше решение, конечно, может использовать любую другую стратегию.

Задача F. Поиск уникального элемента

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

А вы слышали о такой задаче: «Дан массив чисел, в котором все числа кроме одного, встречаются два раза, а оставшееся — один раз. Найти это число.» ?

Вам предстоит решить почти такую же. В этой задаче массив чисел отсортирован, содержит все элементы кроме одного по два раза, оставшийся элемент содержит один раз, но он вам не дан! Вместо этого можно по одному запрашивать его элементы.

Протокол взаимодействия

В начале взаимодействия на вход вашей программе будет подано нечётное число n ($1 \leqslant n \leqslant 199\,999$) — длина массива.

После этого вы можете делать два типа запросов.

- «`?` x». Запросить элемент массива на позиции x ($1 \leqslant x \leqslant n$). Разрешено сделать не более 40 таких запросов. При превышении этого лимита решение получит вердикт «`Wrong Answer`».
- «`!` v». Ответить на задачу. Число v должно быть равно единственному элементу массива, который встречается один раз.

В ответ на запрос первого типа будет получена одна строка, в которой содержится соответствующий элемент массива. Это положительное целое число, не превосходящее 10^9.

После выполнения запроса второго типа решение должно корректно завершиться.

Каждый запрос следует выводить на отдельной строке. 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.

Пример

запросы участника	ответы проверяющей программы	массив
? 1 ? 5 ? 3 ! 2	5 1 3 2	1 1 2 3 3

Задача G. Найдите две точки

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

На плоскости загаданы две точки P и Q. Их положение держится в секрете: известно лишь, что они различны и имеют целые координаты от 0 до 10^6 включительно.

Чтобы найти секретные точки, можно задавать вопросы. Каждый вопрос — это какая-то целая точка R на плоскости. В ответ мы узнаём манхэттенское расстояние от R до ближайшей из точек P и Q. Говоря формально, мы получаем в ответ число $\min\{d(R,P), d(R,Q)\}$, где расстояние вычисляется как $d(A,B) = |x_A - x_B| + |y_A - y_B|$.

Найдите обе секретные точки.

Протокол взаимодействия

В каждом тесте положение двух секретных точек зафиксировано заранее. Координаты секретных точек — целые числа от 0 до 10^6 включительно. Гарантируется, что секретные точки различны.

В каждый момент у решения есть выбор: либо задать вопрос, либо завершить работу.

Чтобы задать вопрос, выведите в отдельной строке координаты точки — два целых числа от -10^9 до $+10^9$ включительно. В ответ решение получит строку с одним целым числом — манхэттенским расстоянием от точки в вопросе до ближайшей из секретных точек. Чтобы найти секретную точку, нужно назвать её в вопросе — и получить в ответ расстояние 0.

Когда решение корректно завершает работу вместо того, чтобы задать очередной вопрос, проверяется, что обе секретные точки найдены хотя бы по разу. Если это не так, решение получает вердикт «`Wrong Answer`».

Прямого ограничения на количество вопросов нет. Разрешается повторять вопросы. Разрешается задавать вопросы после того, как обе секретные точки найдены.

Помните, что после вывода каждого вопроса нужно выводить **перевод строки**, а затем **очищать буфер** вывода: например, функцией «`fflush(stdout)`» в языке C, «`cout.flush()`» в C++ или «`sys.stdout.flush()`» в Python. Кроме того, следует **прочитать ответ** на предыдущий вопрос, прежде чем задавать следующий. В противном случае очередной вопрос может не дойти до проверяющей программы, и тогда решение получит вердикт «`Idleness Limit Exceeded`».

Пример

Пустые строки добавлены для удобства чтения.

стандартный ввод	*стандартный вывод*
4 3 2 0 0 0 7	2 2 4 4 6 2 5 1 5 1 4 7 9 9

Система оценки

В этой задаче две подзадачи. Подзадача считается решённой, когда пройдены все тесты, подходящие под её ограничения.

В первой подзадаче координаты секретных точек — целые числа от 0 до 100 включительно, и за неё можно получить 30 баллов.

Во второй подзадаче координаты секретных точек — целые числа от 0 до 10^6 включительно, и за неё можно получить оставшиеся 70 баллов.

Задача H. Круговой поиск

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

Маленькая Лиза занимается математикой на учебном веб-сайте. Она решила очередную порцию задач, и, как обычно, в качестве вознаграждения на сайте открылась новая игра. В этот раз игра состоит в следующем.

На экране нарисована координатная плоскость, на которой обведён квадрат с противоположными углами в точках $(0, 0)$ и (n, n). Игра загадывает секретную точку с целыми координатами в этом квадрате или на его границе. Задача игрока — её найти. Для этого на плоскость можно поставить круг, выбрав точку с целыми координатами для его центра и неотрицательный целый радиус. После этого игра говорит, находится ли секретная точка в этом круге или на его границе. Игра заканчивается, когда игрок ставит круг радиуса 0 в секретную точку.

Лиза сыграла несколько раз, и после длинной партии задумалась: как быстро найти секретную точку, если n велико?

Решите задачу Лизы в общем виде. Зная число n и ставя на плоскость круги, найдите секретную точку достаточно быстро.

Протокол взаимодействия

Сначала на вход подаётся целое число n в отдельной строке ($1 \leqslant n \leqslant 10\,000$). Далее игрок ставит от 0 до 50 кругов.

Чтобы поставить очередной круг, в отдельной строке выведите три целых числа: «x y r». Здесь x и y — координаты центра круга, а r — его радиус ($-10^9 \leqslant x, y \leqslant +10^9$, $0 \leqslant r \leqslant 10^9$). В ответ игрок получает строку с одним словом: «`Yes`», если секретная точка находится внутри или на границе круга, и «`No`» в противном случае.

После получения «`Yes`» с кругом радиуса 0 следует корректно завершить работу программы. Если же после 50 поставленных кругов этого не произошло, система пытается завершить проверку с вердиктом «`Wrong Answer`».

Чтобы предотвратить буферизацию вывода, после вывода каждой 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.

В каждом тесте координаты точки **зафиксированы заранее** и не меняются по ходу проверки.

Пример

стандартный ввод	*стандартный вывод*	*секретная точка*
1 No Yes Yes	 0 0 1 1 1 1 1 1 0	$(1, 1)$

Задача I. Поиск на зигзаге

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

Есть секретный массив, состоящий из n различных чисел. Позиции в массиве пронумерованы подряд целыми числами от 1 до n. Можно задавать вопрос: чему равно число на позиции i?

Даны k различных чисел из массива. Определите позиции этих k чисел в массиве, задав не более 500 вопросов.

Важное дополнение: массив записан на ленте, концы которой склеены так, что после последнего элемента следует первый. Зацикленный массив устроен как *зигзаг*. А именно, в нём есть две различные позиции, p и q. На части ленты от позиции p до позиции q следующий элемент больше предыдущего. На части ленты от позиции q до позиции p следующий элемент меньше предыдущего. Напомним, что все числа в массиве различны, поэтому все неравенства строгие.

Формат входных данных

В первой строке заданы два целых числа n и k: размер массива и количество чисел для поиска ($2 \leqslant n \leqslant 200\,000$; $1 \leqslant k \leqslant 10$). Во второй строке заданы k различных чисел из массива, позиции которых нужно определить.

Массив в каждом тесте зафиксирован заранее и не меняется в процессе работы.

Протокол взаимодействия

Чтобы узнать число на позиции i ($1 \leqslant i \leqslant n$), выведите номер позиции в отдельной строке — и очистите буфер вывода, чтобы не получить вердикт `IL` (`Idleness Limit Exceeded`). После этого прочитайте в отдельной строке число a_i — i-й элемент массива ($1 \leqslant a_i \leqslant 10^9$, все a_i различны).

Чтобы закончить взаимодействие, выведите в отдельной строке число 0 и завершите работу программы. Решение считается верным, если вопросов (не считая последний 0) было не более 500 и каждое из k заданных во вводе чисел встретилось **хотя бы раз** среди ответов на вопросы.

Заметим, что ограничение на количество вопросов проверяется только после окончания взаимодействия.

Пример

В примере секретный массив таков: $a = (6, 9, 8, 5, 3, 1, 2)$. От позиции $p = 6$ до позиции $q = 2$ следующий элемент больше предыдущего: $1 < 2 < 6 < 9$. От позиции $q = 2$ до позиции $p = 6$ следующий элемент меньше предыдущего: $9 > 8 > 5 > 3 > 1$.

стандартный ввод	*стандартный вывод*	*пояснение*
7 3 1 6 5 5 1 9 6 1	4 6 2 1 6 0	a[4] = 5 (found 5) a[6] = 1 (found 1) a[2] = 9 (miss) a[1] = 6 (found 6) a[6] = 1 (repeat)

Задача J. Режим дня у гномов

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

В маленьком домике на опушке леса живут Белоснежка и n гномов.

Известно, что каждый гном непрерывно спит ровно половину суток, и эта половина начинается всегда в одно и то же время суток. Всё остальное время гном непрерывно бодрствует.

Будучи хозяйкой дома, Белоснежка хочет узнать про каждого гнома, в какое время суток (с точностью до минуты) тот ложится спать. В каждую из 1440 минут в сутках Белоснежка может проверить кровати любых гномов и узнать, спят те или бодрствуют. Но Белоснежка может проверить кровать каждого гнома не больше 50 раз — иначе гном будет возмущён вторжением в его личную жизнь.

Помогите Белоснежке за одни сутки узнать про каждого гнома, в какое время суток (с точностью до минуты) он ложится спать.

В каждом тесте расписание гномов зафиксировано заранее и не меняется в процессе взаимодействия. Другими словами, интерактор в этой задаче не адаптивный.

Протокол взаимодействия

Сначала прочитайте отдельную строку, содержащую целое число n — сколько гномов живёт в домике ($1 \leqslant n \leqslant 100$).

Белоснежка использует часы с 24-часовым циферблатом. Взаимодействие начинается в `00:00` и заканчивается в `23:59`. Чтобы в момент `HH:MM` (часы от `00` до `23`, минуты от `00` до `59`) проверить, спит ли i-й гном, выведите отдельную строку вида «`at HH:MM check` i». В ответ вы получите отдельную строку: «`asleep`», если в эту минуту i-й гном спит, или «`awake`», если он бодрствует. Каждый следующий запрос должен происходить в момент не раньше предыдущего.

Чтобы вывести ответ, выведите отдельную строку «`answer`», а за ней n отдельных строк вида `HH:MM` — время, когда ложатся спать первый, второй, ..., n-й гномы. После этого завершите работу программы.

Если решение выведет слишком много проверок для какого-то гнома или неправильный ответ, а после этого сразу завершит работу, оно получит вердикт `WA` (`Wrong Answer`). Не забывайте после каждой проверки и после ответа выводить перевод строки и очищать буфер вывода, чтобы не получить вердикт `IL` (`Idleness Limit Exceeded`).

Пример

стандартный ввод	*стандартный вывод*
2 asleep awake asleep awake awake	at 01:40 check 1 at 01:40 check 2 at 07:59 check 1 at 08:00 check 1 at 13:41 check 2 answer 20:00 01:41

Пояснение к примеру

В примере в домике живут $n = 2$ гнома.

Первый гном спит с `20:00` до `07:59` включительно. Мы знаем это, потому что в `07:59` он спал, а в `08:00` уже бодрствовал. Значит, гном проснулся ровно в `08:00`. А значит, он ложится спать ровно через 12 часов после этого.

Второй гном спит с `01:41` до `13:40` включительно. Мы знаем это, потому он бодрствовал и в `01:40`, и в `13:41`. С минуты `13:41` до минуты `01:40` включительно проходит ровно половина суток. Значит, это его первая и последняя минуты бодрствования.

Отметим, что в одну и ту же минуту (`01:40` в примере) Белоснежка может проверить кровати нескольких гномов.

Пустые строки добавлены в пример лишь для удобства чтения. В настоящем вводе и выводе пустых строк нет.