Задача A. Поиск буквы «a»

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

Это интерактивная задача.

Есть строка из $2n$ букв, которая зафиксирована в каждом тесте, но держится в секрете. Известно, что в ней n букв «a» и n букв «b».

Нужно найти хотя бы одну букву «a» в этой строке. Для этого можно задавать вопросы. Каждый вопрос имеет вид «какая буква находится на позиции x?». Если это буква «a», следует сразу завершить работу программы. Если же это буква «b», придётся задать ещё вопрос.

Напишите программу, которая находит букву «a» не более чем за 100 вопросов.

Протокол взаимодействия

В первой строке ввода задано целое число n ($1 \leqslant n \leqslant 100\,000$).

Чтобы задать вопрос «какая буква находится на позиции x?», выведите целое число x на отдельной строке ($1 \leqslant x \leqslant 2 \cdot n$). 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.

В ответ на каждый вопрос во вводе появится новая строка, содержащая одну букву английского алфавита: «a» или «b».

Если в ответ на вопрос получена буква «a», следует сразу завершить работу программы.

Если после 100 вопросов ни одна буква «a» так и не найдена, проверка завершается с вердиктом «`Wrong Answer`».

Пример

стандартный ввод	*стандартный вывод*
1 b a	2 1

Задача B. Линейный конгруэнтный генератор

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

Как известно, линейный конгруэнтный генератор псевдослучайных чисел устроен следующим образом. Зафиксированы числа a, c и m. Кроме того, у генератора есть состояние s. Когда нужно сгенерировать очередное псевдослучайное число, происходит присваивание $s_{\text{new}} \leftarrow (s_{\text{old}} \cdot a + c) \bmod m$, после чего новое значение s объявляется ответом.

В этой задаче $m = 2^{32}$, а числа a, c и начальное состояние s заданы во входных данных. Выведите следующие 10 чисел, которые сгенерирует этот генератор.

Формат входных данных

В первой строке записаны через пробел три целых числа: a, c и s. Гарантируется, что эти числа помещаются в 32-битный беззнаковый тип данных.

Формат выходных данных

Выведите 10 следующих чисел, которые сгенерирует получившийся генератор.

Пример

стандартный ввод
69069 1 12345
стандартный вывод
852656806 3856338159 1023442532 1580485141 1639408594 4089354539 1989334640 1055483825 2732393918 2852536103

Задача C. Случайные тоннели

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

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

Пояснение к примеру

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. Многочлен в чёрном ящике

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

Это интерактивная задача.

У Алисы есть чёрный ящик, работающий с целыми числами по модулю $m = 10^9 + 7$. На клавиатуре ящика можно набрать число x, и тогда на экране появится число, равное значению многочлена $p(x) = (a_d x^d + a_{d-1} x^{d-1} + \ldots + a_1 x^1 + a_0) \bmod m$. Степень многочлена d, как и его коэффициенты a_i, неизвестны. Известно только, что $0 \leqslant d \leqslant 10$ и $a_d \neq 0$.

Алиса может ввести несколько чисел x и узнать значения многочлена для этих чисел. Помогите ей узнать степень многочлена d. Вводить числа x можно не больше $d + 3$ раз.

Протокол взаимодействия

Чтобы узнать значение многочлена для числа x, выведите строку вида «`ask` x» ($0 \leqslant x < 10^9+7$). В ответ вы получите строку со значением $p(x)$ — или, если таких вопросов было больше, чем $d + 3$, вместо значения вы получите число -1, после чего проверка завершится.

Чтобы выдать ответ, выведите строку вида «`degree` d». После этого следует корректно завершить работу программы.

После вывода каждой строки следует очищать буфер вывода, иначе вы получите вердикт `Idleness Limit Exceeded`: 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.

Пример

стандартный ввод	*стандартный вывод*
1000000006 7 34 98	ask 1 ask 3 ask 6 ask 10 degree 2

Замечание

В каждом тесте степень и коэффициенты многочлена $p(x)$ выбраны и зафиксированы заранее.

В примере, который заодно является первым тестом при проверке, $p(x) = x^2 + 1\,000\,000\,005$. При создании всех остальных тестов была выбрана степень d ($0 \leqslant d \leqslant 10$), после чего в качестве $p(x)$ был случайным образом равновероятно выбран один из многочленов такой степени.

Задача E. Партии

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	4 секунды
Ограничение по памяти:	512 мебибайт

Если правящим родам в Вестеросе что-то нужно, они объединяются в партии. Все атрибуты — программа партии, зачем она была создана — давно канули в Лету, остались лишь по инерции плывущие старые союзы.

Однажды возникнув, партия, в силу традиций, никогда не распадается. В какой-то момент партий стало так много, что новые партии стали образовываться только из старых.

Если две партии хотят *образовать* новую партию, то в неё вступают только те рода, которые были **ровно в одной из двух** партий. Заметим, что в итоге партия может получиться пустой.

Вам предоставлены летописи, состоящие из списков первоначальных партий, оставшихся с древних времён, и описания того, как образовывались новые партии. Все записи даны в хронологическом порядке. Изначально существует N партий с номерами от 1 до N. При создании новой партии ей выдаётся минимально возможный из ещё не занятых номеров (все номера, разумеется, должны являться натуральными числами).

Одна партия считается такой же, как другая, если два множества родов, которые состоят в этих двух партиях, совпадают. Для каждой новой образовавшейся партии вас просят сообщить, сколько таких же партий, как эта, было на момент её создания.

Формат входных данных

В первой строке заданы два натуральных числа N и M — количество первоначальных партий и количество объединений соответственно. Оба числа не превосходят одного миллиона.

Далее в N строках описаны составы первоначальных партий. Описание каждой партии начинается с числа k — количества родов в партии ($k \geqslant 1$). Далее следует k чисел a_1, a_2, …, a_k — номера родов, присутвующих в партии. Гарантируется, что все a_i в описании одной партии попарно различны и $1 \leqslant a_i \leqslant 10^6$. Также гарантируется, что сумма всех значений k не превосходит 10^6.

Далее следует M строк. Каждая из них содержит два числа x_i и y_i — номера партий, из которых образовалась новая партия. Гарантируется, что партии с такими номерами уже существуют к этому моменту.

Формат выходных данных

Выведите M чисел, разделённых пробелами. Для каждой из M новых партий выведите, сколько таких же партий, как она, существовало к моменту её создания.

Пример

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

Пояснение к примеру

Изначально было три партии — $\{1\}$, $\{2\}$, $\{1, 2\}$. Потом партии с номерами 1 и 2 образовали новую партию $\{1, 2\}$ с номером 4 — такую же, как партия с партией 3. Далее партии 3 и 4 образовали партию с номером 5, которая получилась пустой (обратите внимание, что это тоже считается партией!). Затем партии 3 и 5 образовали партию с номером 6, которая совпадает по составу с партиями 3 и 4. Наконец, образованная партиями 3 и 1 партия с номером 7 получилась такой же, как партия 2.

Задача F. Циклический сдвиг или нет

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

Это задача с двойным запуском.

Дана строка s из n двоичных цифр. Также дана ещё одна строка t, содержащая n двоичных цифр. Выясните, является ли одна строка циклическим сдвигом другой.

Строка s является циклическим сдвигом t, если существует число k ($0 \leqslant k < n$), для которого $s_1 s_2 \dots s_n = t_{k+1} t_{k+2} t_n t_1 t_2 \dots t_k$.

В чём сложность? — спросите вы. Ваше решение запускается два раза, и строки даны в разных запусках.

Первый запуск

При первом запуске решение получает строку s. В первой строке записано слово «`first`». Во второй строке задано целое число n — длина строки ($1 \leqslant n \leqslant 100\,000$). В третьей строке задана сама строка s, состоящая из n двоичных цифр без пробелов.

Выведите подсказку h — строку содержащую от 1 до 100 произвольных символов с ASCII-кодами от 32 до 126 включительно.

Второй запуск

При втором запуске решение получает строку t и подсказку h. В первой строке записано слово «`second`». Во второй строке задано целое число n — длина строки (та же, что при первом запуске). В третьей строке задана сама строка t, состоящая из n двоичных цифр без пробелов. Наконец, в четвёртой строке дана подсказка h, выведенная при первом запуске.

Выведите «`YES`», если t является циклическим сдвигом s, и «`NO`» в противном случае. Каждую букву можно выводить в любом регистре.

В каждом тесте строки s и t зафиксированы заранее.

Пример

В каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте.

стандартный ввод	*стандартный вывод*

`first`
`6`
`011001`
	`#@! 001011 !@#`

`second`
`6`
`100101`
`#@! 001011 !@#`
	`YES`

Задача G. Передача дежурства

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

Аня сегодня дежурит в лаборатории. Там у дежурного на пульте есть — без преувеличения — целый миллион переключателей! Переключатели пронумерованы целыми числами от 1 до 10^6, и каждый переключатель отвечает за прибор с таким же номером. По переключателям непонятно, включены их приборы или выключены, но известно, что нажатие на переключатель меняет состояние со включённого на выключенное и наоборот.

Утром, когда Аня приходит на дежурство, все приборы выключены. Затем приходят другие сотрудники и нажимают на переключатели.

Чтобы оптимизировать потребление энергии, после каждого переключения дежурный должен различать следующие классы состояний:

- все приборы выключены,
- включён ровно один прибор — нужно знать, какой именно,
- включено два или больше приборов.

Аня, конечно, справится с этим заданием. Но после неё будет дежурить её друг Андрей. А при передаче дежурства Аня может оставить Андрею лишь короткую записку. После прочтения Андрей будет дежурить точно так же, как Аня: другие сотрудники будут нажимать на переключатели, а ему нужно будет знать, в каком из классов состояний находится лаборатория.

Помогите ребятам придумать, что написать в записке, чтобы не только Аня, но и Андрей после каждого переключения обладал всей нужной информацией.

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза. В каждом тесте все нажатия при обоих запусках зафиксированы заранее. В конце каждой строки входных данных следует символ перевода строки.

Первый запуск

При первом запуске решение действует за Аню. В первой строке записано слово «`start`». Во второй строке записано целое число n — количество переключений ($1 \leqslant n \leqslant 100\,000$). Каждая из следующих n строк содержит одно целое число — номер прибора (от 1 до 10^6), на переключатель которого нажал сотрудник.

В ответ на каждое переключение выведите строку с одним числом:

- `0`, если все приборы выключены,
- номер включённого прибора, если включён ровно один прибор,
- `-1`, если включено два или больше приборов.

После всех ответов выведите в отдельной строке записку, которую Аня оставит Андрею. Записка должна иметь длину от 0 до 1000 символов и состоять из символов с ASCII-кодами от 32 до 126. Никаких других ограничений на содержимое записки нет.

Второй запуск

При втором запуске решение действует за Андрея. В первой строке записано слово «`resume`». Во второй строке дана записка — ровно то, что решение вывело при первом запуске. В третьей строке записано целое число m — количество переключений ($1 \leqslant m \leqslant 100\,000$). Каждая из следующих m строк содержит одно число — номер прибора (от 1 до 10^6), на переключатель которого нажал сотрудник.

В ответ на каждое переключение выведите строку с одним числом — по тем же правилам, что и при первом запуске.

Пример

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте.

стандартный ввод	*стандартный вывод*
start 5 10 14 10 12 10	10 -1 14 -1 -1 3 10 12 14

стандартный ввод	*стандартный вывод*
resume 3 10 12 14 6 14 277 12 10 277 12	-1 -1 -1 277 0 12

Задача H. 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 I. 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 J. Два пропавших числа

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

У Глаши была последовательность из $n \geqslant 4$ элементов: каждое целое число от 1 до n встречалось в ней ровно один раз. Глаша взяла эту последовательность и стёрла из неё два последних элемента. Потом она разделила оставшуюся последовательность на две непустые части — левую и правую. В понедельник Глаша написала на доске левую часть последовательности, а во вторник — правую часть.

Гриша хочет узнать, какие числа Глаша стёрла. В понедельник он смотрит на левую часть последовательности, но может запомнить ко вторнику не очень большое количество информации. Во вторник Гриша видит правую часть последовательности, а также запомненную информацию с понедельника.

Напишите программу, которая поможет Грише узнать стёртые числа.

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза. В каждом тесте все действия Глаши зафиксированы заранее. В конце каждой строки входных данных следует символ перевода строки.

Первый запуск

При первом запуске решение получает левую часть последовательности. В первой строке записано слово «`first`». Во второй строке записаны целые числа n и k_1 — исходная длина последовательности и длина левой части ($4 \leqslant n \leqslant 300\,000$, $0 < k_1 < n - 2$). В третьей строке записаны через пробел k_1 чисел — левая часть последовательности (все числа целые от 1 до n, все они различны).

Выведите в отдельной строке памятку — информацию, которую Гриша должен запомнить до вторника. Эта памятка должна иметь длину от 0 до 1000 символов и состоять из символов с ASCII-кодами от 32 до 126. Никаких других ограничений на содержимое памятки нет.

Второй запуск

При втором запуске решение получает памятку с понедельника, а также правую часть последовательности. В первой строке записано слово «`second`». Во второй строке дана памятка — ровно то, что решение вывело при первом запуске. В третьей строке записаны целые числа n и k_2 — исходная длина последовательности и длина правой части (n такое же, как при первом запуске, $0 < k_2 < n$ и $k_1 + k_2 = n - 2$). В четвёртой строке записаны через пробел k_2 чисел — правая часть последовательности (все числа целые от 1 до n, все они различны и отличаются от чисел в левой части последовательности).

Выведите в отдельной строке два числа — те целые числа от 1 до n, которых нет ни в левой, ни в правой части последовательности. Числа можно выводить в любом порядке.

Пример

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте.

стандартный ввод	*стандартный вывод*

`first`
`9 3`
`4 6 1`
	`3  4 6 1`

стандартный ввод	*стандартный вывод*

`second`
`3  4 6 1`
`9 4`
`5 2 9 8`
	`7 3`

Задача K. Поиск маленьких чисел

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

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 $k \leqslant 10$.

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 выходных данных

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

Пояснение к примеру

В примере загадана перестановка `1 2 3`.

Задача L. Кольцевой маршрут

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

Это задача с двойным запуском.

В подземной стране гоблинов $n \leqslant 50\,000$ городов, пронумерованных числами от 1 до n. Чтобы быстро перемещаться между городами, гоблины построили одну кольцевую ветку метро, проходящую через каждый город по одному разу. Все поезда едут по этому кольцу в одну и ту же сторону.

Фея Маргарет хочет узнать, в каком порядке поезд посещает города. Для этого она может послать в подземную страну $k \leqslant 1000$ туристов-гномиков. Каждому гномику Маргарет укажет начальный город. Гномик сядет в поезд в этом городе, запишет по порядку номера первых $\ell \leqslant 1000$ городов, которые он посетит, включая начальный город, после чего выйдет из поезда, поднимется на поверхность и вернётся к фее. Число ℓ одно и то же для всех гномиков.

Помогите Маргарет узнать полный маршрут поезда и запишите его, начиная с города 1.

Первый запуск

При первом запуске решение выбирает количество гномиков, длину их маршрутов и номера начальных городов.

В первой строке записано слово «`ask`». Во второй строке задано целое число n — количество городов в подземной стране ($1 \leqslant n \leqslant 50\,000$).

В первой строке выведите два целых числа: количество гномиков k и длину маршрута ℓ ($1 \leqslant k, \ell \leqslant 1000$). Во второй строке выведите k чисел: номера начальных городов для k гномиков.

Второй запуск

При втором запуске решение получает маршруты гномиков, после чего должно восстановить полный маршрут поезда.

В первой строке записано слово «`answer`». Во второй строке заданы два целых числа: количество гномиков k и длина маршрута ℓ (те, что выведены при первом запуске). Каждая из следующих k строк содержит ℓ целых чисел: города, которые посетил очередной гномик, указанные в порядке посещения.

Выведите строку, содержащую n целых чисел: полный маршрут поезда. Маршрут должен начинаться в городе с номером 1.

Замечание

В каждом тесте маршрут поезда зафиксирован заранее, но держится в секрете.

Пример

В каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте.

стандартный ввод	*стандартный вывод*
`ask` `5`	`2 3` `1 2`

`answer`
`2 3`
`1 3 5`
`2 1 3`
	`1 3 5 4 2`

Задача M. Прямоугольник из хаоса

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

Даны n точек на плоскости. Координаты каждой точки — целые числа от 0 до 10^5 включительно. Каждая точка выбрана случайно, равновероятно среди всех возможных точек и независимо от остальных; в частности, точки могут совпадать.

Выберем из данных точек четыре точки в вершинах прямоугольника с горизонтальными и вертикальными сторонами. Вычислим площадь этого прямоугольника. Какое максимальное число у нас может получиться?

Формат входных данных

В первой строке задано целое число n — количество точек ($1 \leqslant n \leqslant 300\,000$). Каждая из следующих n строк содержит два целых числа x_i и y_i — координаты очередной точки ($0 \leqslant x_i, y_i \leqslant 10^5$).

Формат выходных данных

Выведите максимальную площадь прямоугольника с горизонтальными и вертикальными сторонами, у которого все четыре вершины находятся в заданных точках. Если такого прямоугольника нет, выведите 0.

Пример

стандартный ввод	*стандартный вывод*
2 52001 45162 73830 41780	0

Задача N. Умножение

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

Это интерактивная задача.

Жюри выбрало секретное нечётное число x от 1 до $2^{31} - 1$ включительно. Ваша задача — угадать его. Для этого жюри даёт вам чётное число n. После этого вы должны вывести **ровно** n различных целых чисел от 0 до $2^{31} - 1$ включительно. Далее жюри умножит все эти числа на x и возьмёт результаты умножения по модулю 2^{31}. Потом жюри равновероятно выберет случайное подмножество получившихся чисел размера $n/2$ и даст его элементы вам в случайном порядке. В ответ вы должны будете вывести x.

В каждом тесте число x выбрано заранее и не меняется.

Протокол взаимодействия

Исходно вам даётся одно чётное число n ($4 \leqslant n \leqslant 10^5$). После этого вы должны вывести n различных целых чисел $a_1, a_2, \ldots, a_n$ ($0 \leqslant a_i \leqslant 2^{31} - 1$) на одной строке через пробел. Далее, вам даются $n/2$ целых чисел $b_1, b_2, \ldots, b_{n/2}$ ($0 \leqslant b_i \leqslant 2^{31} - 1$), полученные описанным выше способом, на одной строке через пробел. Наконец, вы должны вывести нечётное число x: секретное число, загаданное жюри ($1 \leqslant x \leqslant 2^{31} - 1$).

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. А ещё не забывайте выводить перевод строки в конце каждой выведенной строки.

Пример

стандартный ввод	*стандартный вывод*
4 9 6	 1 2 3 4 3

Задача O. Удвоение прямоугольников

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

Гриша и Дима играют в *удвоение прямоугольников*. Поле для этой игры представляет собой полоску, разделённую на w равных клеток, пронумерованных натуральными числами от 1 до w.

Изначально Гриша владеет клеткой x, а Дима — клеткой y. Назовём эти клетки прямоугольниками размера 1. Соперники ходят по очереди; Гриша ходит первым. На своём ходу игрок берёт имеющийся у него прямоугольник и удваивает его либо вправо, либо влево. При этом прямоугольник после удвоения не должен выходить за границы полоски.

Пусть, к примеру, игра ведётся на полоске из 5 клеток, и у кого-то сейчас есть прямоугольник, покрывающий клетки $[2..3]$ (включительно). Тогда его можно удвоить вправо и получить прямоугольник $[2..5]$, а вот влево, увы, нельзя, так как прямоугольник $[0..3]$ захватывает клетку с номером 0, которой нет на полоске.

Победителем считается игрок, после чьего хода пересечение прямоугольников впервые стало **непустым**, то есть у прямоугольников Гриши и Димы появилась общая клетка.

По заданной длине полоски и описанию стартовых позиций определите, кто из ребят выиграет. Можно показать, что игра не может закончиться вничью.

Формат входных данных

В первой строке задан размер поля w ($2 \leqslant w \leqslant 10^5$).

Во второй строке через пробел следуют два числа x и y — номера клеток, изначально занятых Гришей и Димой ($1 \leqslant x < y \leqslant w$).

Формат выходных данных

В случае победы Гриши выведите «`letoucan`»; в противном случае выведите «`cdkrot`».

Выводите ответ без кавычек.

Примеры

стандартный ввод	*стандартный вывод*
4 1 4	letoucan
4 2 3	letoucan
4 1 3	cdkrot

Пояснения к примерам

Промоделируем первый пример.

Изначально Гриша занимает клетку 1, а Дима — клетку 4. Расширяться влево Гриша не может, так как он выйдет за границы полоски (по аналогичной причине Дима не может расширяться вправо).

Тогда состояния последовательно изменяются следующим образом:

- Первый ход Гриши: $\big([1], [4]\big) \to \big([1..2], [4]\big)$;
- Первый ход Димы: $\big([1..2], [4]\big) \to \big([1..2], [3..4]\big)$;
- Второй ход Гриши: $\big([1..2], [3..4]\big) \to \big([1..4], [3..4]\big)$.

После этого хода у прямоугольника Гриши появились общие клетки с прямоугольником Димы, что означает победу Гриши.

Во втором примере Гриша выигрывает первым же ходом — ему достаточно удвоить свой прямоугольник вправо.