

## Задача А. Поиск буквы «а»

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

*Это интерактивная задача.*

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

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

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

### Протокол взаимодействия

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

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

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

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

### Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>
1	2
b	1
a	

## Задача В. Линейный конгруэнтный генератор

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

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

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

## Задача С. Случайные тоннели

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

*Это интерактивная задача.*

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

Когда демиург создавал Туманный Путь, он задал вероятность  $p$  того, что для любой пары целых чисел  $(i, j)$  от звезды  $i$  есть прямой туннель к звезде  $j$ , а далее предоставил конструирование туннелей воле случая. Вероятность существования каждого туннеля не зависит от вероятности существования любых других туннелей. Все туннели односторонние, то есть при  $i \neq j$  туннель от  $i$  к  $j$  и туннель от  $j$  к  $i$  — это два разных туннеля, 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 туннелю к звезде  $x$ », где  $x$  — целое число от 0 до  $2^{32} - 1$ . После такого запроса, если существует туннель 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 туннеля.

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 удастся. 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 туннель.

### 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 туннеля в нём равна 50%. Прямого туннеля от звезды 0 к звезде 1 нет. Зато удастся 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 удастся 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 мегабайт

*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  $p(x) = (a_d x^d + a_{d-1} x^{d-1} + \dots + a_1 x^1 + a_0) \bmod m$ . Степень многочлена  $d$ , как и его коэффициенты  $a_i$ , неизвестны. Известно только, что  $0 \leq d \leq 10$  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  $x$ » ( $0 \leq x < 10^9 + 7$ ). 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

<i>стандартный ввод</i>	<i>стандартный вывод</i>
1000000006	ask 1
7	ask 3
34	ask 6
98	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,  $p(x) = x^2 + 1000000005$ . При создании всех остальных тестов была выбрана степень  $d$  ( $0 \leq d \leq 10$ ), после чего в качестве  $p(x)$  был случайным образом равномерно выбран один из многочленов такой степени.

## Задача Е. Проверка равенства

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

Маленький Джошуа учится умножать матрицы. Он тренируется, умножая большие бинарные матрицы в  $\mathbb{F}_2$  (все арифметические операции производятся по модулю два). Недавно он перемножил две матрицы  $A$  и  $B$  размера  $n \times n$  и получил результат  $C$ .

Дженни не верит ему и говорит, что Джошуа ошибся. Помогите детям понять, кто прав. Даны матрицы  $A$ ,  $B$ ,  $C$ . Проверьте, что  $AB = C$ .

### Формат входных данных

В первой строке задано целое число  $n$  — размер матриц ( $1 \leq n \leq 4000$ ). Следующие три строки содержат описания матриц  $A$ ,  $B$  и  $C$ .

Каждая матрица описывается строкой, содержащей  $n$  блоков размера  $\lceil n/4 \rceil$  16-ричных цифр. Если записать цифры в двоичной записи в данном порядке от старших цифр к младшим и обрезать лишние цифры в конце строки, получится очередная строка матрицы. Например, матрица

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

описывается как 28, A8, 68, 78, D0, 88.

### Формат выходных данных

Выведите "YES", если  $AB = C$ , иначе "NO".

## Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>
6 28 A8 68 78 D0 88 80 40 20 10 08 04 28 A8 68 78 D0 88	YES
6 28 A8 68 78 D0 88 80 40 20 10 08 04 28 B8 68 78 D0 88	NO

## Задача F. Степень матрицы

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

Вам дана квадратная матрица  $A$ , состоящая из целых чисел. Найдите наименьшее целое положительное число  $k$ , для которого выполняется равенство  $A^k = 0 \pmod{10^9 + 7}$ . Другими словами, в матрице  $A^k$  все элементы должны делиться нацело на  $10^9 + 7$ .

### Формат входных данных

В первой строке дано число  $n$  ( $1 \leq n \leq 300$ ).

В следующих  $n$  строках задана матрица  $A$  размера  $n \times n$ : в  $i$ -й строке  $j$ -е число равно  $A_{ij}$  ( $0 \leq A_{ij} < 10^9 + 7$ ).

### Формат выходных данных

Выведите  $k$ . Если такого  $k$  не существует, то выведите «Infinity» (без кавычек).

### Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>
3 0 1 0 0 0 1 0 0 0	3

## Задача G. Партии

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

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 ( $k \geq 1$ ). Далее следует  $k$  чисел  $a_1, a_2, \dots, a_k$  — номера родов, присутствующих в партии. Гарантируется, что все  $a_i$  в описании одной партии попарно различны и  $1 \leq a_i \leq 10^6$ . 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

<i>стандартный ввод</i>	<i>стандартный вывод</i>
3 4	1 0 2 1
1 1	
1 2	
2 1 2	
1 2	
3 4	
3 5	
3 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 Н. Передача дежурства

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

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 ( $1 \leq n \leq 100\,000$ ). 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 ( $1 \leq m \leq 100\,000$ ). 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.

<i>стандартный ввод</i>	<i>стандартный вывод</i>
start	10
5	-1
10	14
14	-1
10	-1
12	3 10 12 14
10	

<i>стандартный ввод</i>	<i>стандартный вывод</i>
resume	-1
3 10 12 14	-1
6	-1
14	277
277	0
12	12
10	
277	
12	

## Задача I. Поиск уникального элемента

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

*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  $n$  ( $1 \leq n \leq 199\,999$ ) — длина массива.

После этого вы можете делать два типа запросов.

- «?  $x$ ». Запросить элемент массива на позиции  $x$  ( $1 \leq x \leq n$ ). 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	1 1 2 3 3
? 5	1	
? 3	3	
! 2	2	

## Задача J. Два пропавших числа

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

У Глаши была последовательность из  $n \geq 4$  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 ( $4 \leq n \leq 300\,000$ ,  $0 < k_1 < n - 2$ ). 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 выве-

ло 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.

<i>стандартный ввод</i>	<i>стандартный вывод</i>
first	3 4 6 1
9 3	
4 6 1	

<i>стандартный ввод</i>	<i>стандартный вывод</i>
second	7 3
3 4 6 1	
9 4	
5 2 9 8	

## Задача К. Поиск маленьких чисел

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

*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  $k \leq 10$ .

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

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

### 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 мегабайт

*Это задача с двойным запуском.*

В подземной стране гоблинов  $n \leq 50\,000$  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  $k \leq 1000$  туристов-гномиков. Каждому гномику Маргарет укажет начальный город. Гномик сядет в поезд в этом городе, запишет по порядку номера первых  $\ell \leq 1000$  городов, которые он посетит, включая начальный город, после чего выйдет из поезда, поднимется на поверхность и вернётся к фее. Число  $\ell$  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 ( $1 \leq n \leq 50\,000$ ).

В первой строке выведите два целых числа: количество гномиков  $k$  и длину маршрута  $\ell$  ( $1 \leq k, \ell \leq 1000$ ). 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  $\ell$  (те, что выведены при первом запуске). Каждая из следующих  $k$  строк содержит  $\ell$  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.

<i>стандартный ввод</i>	<i>стандартный вывод</i>
ask	2 3
5	1 2
answer	1 3 5 4 2
2 3	
1 3 5	
2 1 3	

## Задача М. Прямоугольник из хаоса

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

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 ( $1 \leq n \leq 300\,000$ ). Каждая из следующих  $n$  строк содержит два целых числа  $x_i$  и  $y_i$  — координаты очередной точки ( $0 \leq x_i, y_i \leq 10^5$ ).

### 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

<i>стандартный ввод</i>	<i>стандартный вывод</i>
2 52001 45162 73830 41780	0

## Задача N. Интерактивная простота

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

*Это интерактивная задача.*

Жюри загадало  $T \leq 10$  целых чисел от 1 до  $10^{18}$  включительно. Вам надо угадать все эти числа одно за другим.

Пусть жюри загадало число  $x$ . Вы можете несколько раз назвать целое число  $y$  от 1 до  $10^{18}$  включительно, в ответ вам сообщат, является ли число  $x + y$  простым или составным. Когда вы угадаете число, назовите его, после чего начните угадывать следующее.

Во всех тестах, кроме примера, все  $T$  чисел выбраны случайно, равномерно и независимо из отрезка  $[1; 10^{18}]$ . Вам требуется угадать их все, задав суммарно не более 8750 вопросов жюри.

### Протокол взаимодействия

Для начала прочитайте строку, содержащую целое число  $T$  — количество загаданных жюри чисел ( $1 \leq T \leq 10$ ). Жюри загадало  $T$  целых чисел  $x_1, \dots, x_T$ , и вам требуется по очереди угадать их все ( $1 \leq x_i \leq 10^{18}$ ).

Пусть сейчас происходит угадывание числа  $x_i$ . Чтобы задать вопрос, выведите в отдельной строке «?  $y$ » ( $1 \leq y \leq 10^{18}$ ). Далее прочитайте строку:

- если  $x_i + y$  простое, то вы прочитаете слово «Prime»;
- если  $x_i + y$  составное, то вы прочитаете слово «Composite»;
- если вы задали некорректный вопрос (8751-й за всё время работы программы; или не подходящий под указанный выше формат; или в котором  $y$  не является целым положительным числом в требуемых пределах), то вы прочитаете слово «Busted».

Когда вы считаете, что  $x_i = z$ , выведите в отдельной строке «!  $z$ » ( $1 \leq z \leq 10^{18}$ ). Эта строка не считается вопросом. Далее прочитайте строку:

- если действительно  $x_i = z$ , то вы прочитаете слово «Correct». Если  $i = T$ , программа должна завершиться; иначе перейдите к угадыванию числа  $x_{i+1}$ .
- если  $x_i \neq z$  или если вы нарушили формат вывода, то вы прочитаете слово «Busted».

Прочитав «Busted», ваша программа должна сразу завершить работу, чтобы получить вердикт Wrong Answer. В противном случае вердикт может быть каким угодно.

### Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>
1	? 6
1 1	? 5
	? 3
	? 1
	! 2

### Замечание

В задаче ровно 30 тестов. В  $i$ -м тесте  $T = \min\{i, 10\}$ . В каждом тесте  $T$  секретных чисел загаданы заранее: они одни и те же для всех запусков всех решений и не меняются во время работы.

## Задача О. Археология

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

*Это интерактивная задача.*

В одной из  $N^2$  целочисленных точек квадрата  $\mathbb{D} = [1; N] \times [1; N]$  спрятан загадочный артефакт. Ваша задача — отыскать этот артефакт. У вас есть радар, его однократное использование устроено так:

- Сначала вы устанавливаете радар в любую целочисленную точку, принадлежащую  $\mathbb{D}$ .
- Если ровно в ней же находится артефакт, радар загорается зелёным, на экране высвечиваются координаты точки, где находятся артефакт и радар, и археологические раскопки на этом оканчиваются.
- В противном случае предположим, что вы установили радар в точку  $R$ , а артефакт находится в точке  $A \neq R$ . Тогда радар загорится жёлтым и отобразит вам координаты такой целочисленной точки  $B \in \mathbb{D}$ , что  $\angle ARB < 90^\circ$ . Из всех таких целочисленных точек радар отобразит равномерно выбранную случайную точку; в частности, если  $A \neq R$ , всегда есть шанс, что радар выдаст координаты точки  $A$  (поскольку  $\angle ARA = 0^\circ < 90^\circ$ ), но он никогда не выдаст точку  $R$ .

Вы можете считать, что, кроме тестов 1–5, из всех возможных точек квадрата  $\mathbb{D}$  артефакт спрятан в равномерно выбранной случайной точке, эта точка зафиксирована в каждом тесте и не зависит от ответов вашей программы. Все описанные выше случайные выборы совершаются независимо.

Ваш радар не полностью заряжен, так что вы сможете использовать его не более 90 раз. Напишите программу, которая будет сообщать, куда ставить радар, чтобы в итоге он загорелся зелёным.

### Протокол взаимодействия

В первой строке находится целое число  $N$  — сторона квадрата, в котором спрятан артефакт ( $1 \leq N \leq 10^9$ ).

Далее ваша программа должна выводить запросы об использовании радара. Каждый запрос состоит из двух целых чисел  $R_{xi}$  и  $R_{yi}$  — координат очередной точки, куда вы хотите поставить радар ( $1 \leq R_{xi}, R_{yi} \leq N$ ). После каждого запроса обязательно выводите перевод строки и сбрасывайте буфер вывода. После этого вы можете считать два целых числа  $B_{xi}$  и  $B_{yi}$  — координаты точки, отобразившиеся на экране радара ( $1 \leq B_{xi}, B_{yi} \leq N$ ). Если

$B_{xi} = R_{xi}$  и  $B_{yi} = R_{yi}$ , то вы успешно прошли тест, и программа должна немедленно завершиться, чтобы получить вердикт ОК на этом тесте. В противном случае, если вы всё ещё не сделали 90 запросов, вы должны продолжать делать запросы.

Ваша программа должна немедленно завершиться, если после совершения 90 запросов радар так и не загорелся зелёным, чтобы получить вердикт Wrong Answer, иначе вердикт может быть непредсказуемым.

### Примеры

<i>стандартный ввод</i>	<i>стандартный вывод</i>
1 1 1	1 1
2 2 2 1 2 1 2 2 2 1 1 1 2 1 1 1 2	1 1 1 1 2 1 2 1 2 1 2 2 2 2 1 2

### Замечание

В этой задаче ровно 50 тестов, включая примеры.

Тест 1 совпадает с первым примером.

В тестах 2–5  $N = 2$ . Артефакт во втором тесте скрыт в точке  $(1, 2)$ , как показано во втором примере; однако не гарантируется, что во втором тесте интерактор будет давать такие же ответы на запросы, как в примере. В тестах 2–5 гарантировано будут представлены все четыре возможных точки, где спрятан артефакт.

В следующих 28 тестах  $N$  последовательно принимает значения  $4, 8, 15, 30, \dots, \frac{10^9}{4}, \frac{10^9}{2}$ . Формально, для каждого  $i \in \{6, \dots, 33\}$ , в  $i$ -м тесте  $N = \lceil 10^9 / 2^{34-i} \rceil$ . В каждом из этих тестов артефакт скрыт в равномерно выбранной случайной точке.

В следующих 17 тестах  $N$  последовательно принимает значения  $10^9 - 5, \dots, 10^9 - 1, 10^9, 10^9, \dots, 10^9$ . Формально, для каждого  $i \in \{34, \dots, 50\}$ , в  $i$ -м тесте  $N = 10^9 + \min\{0, i - 39\}$ . В этих 17 тестах артефакт также каждый раз скрыт в равномерно выбранной случайной точке.