Задача A. Угадай число

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

Это интерактивная задача. В процессе тестирования программа участника взаимодействует с программой жюри с использованием стандартных потоков ввода/вывода.

Программа жюри загадала число от 1 до n, цель программы участника — отгадать его, задав не более чем 30 вопросов. Для этого программа участника сообщает свои догадки программе жюри, а программа жюри отвечает, является ли загаданное число бóльшим, меньшим или равным сделанной догадке.

Протокол взаимодействия

Сначала необходимо прочитать из стандартного потока ввода число n ($1 \leqslant n \leqslant 10^9$). Затем протокол общения следующий: требуется вывести в стандартный поток вывода одну строку, содержащую целое число — свою догадку о загаданном числе.

После этого необходимо считать из стандартного потока ввода одно число: сообщение программы жюри. Возможны следующие сообщения:

- «`1`» — загаданное число больше последней догадки;
- «`-1`» — загаданное число меньше последней догадки;
- «`0`» — последняя догадка верна. Считав 0, ваша программа должна завершиться.

Обратите внимание на необходимость перевода строки после каждой выведенной догадки, прочитайте подробности про интерактивные задачи в памятке участника.

Пример

стандартный ввод	стандартный вывод
5 -1 1 0	 3 1 2

В примере ввод и вывод отформатированы пустыми строками, чтобы было видно, какие запросы соответствуют каким ответам программы жюри. В реальном взаимодействии необходимо переводить строку после каждого запроса, но выводить пустые строки не надо.

Задача B. В поисках истины

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

Это интерактивная задача.

После вывода очередной строки не забывайте очищать буфер потока вывода после каждого запроса. Для этого можно, например, воспользоваться командами fflush(stdout), cout.flush(), либо выполнять перевод строки при помощи endl в C++, system.out.flush() в Java, flush(output) в Pascal или sys.stdout.flush() в Python.

Сарком управлял совет Великих сквайров. В совете состояло пятеро саркитов, и самый богатый и влиятельный из них, сквайр Файф, взял на себя ответственность разобраться в истории космоаналитика. Волей случая к нему попал и Рик, который оказался космоаналитиком, а Селим Джунц и Людиган Эбл из посольства Трантора были готовы с ним сотрудничать. Сквайра интересовало одно — кто же мог совершить столь ужасное преступление?

Еще раньше Великому сквайру поступали анонимные письма, в которых сообщалось, что Флорина должна погибнуть. Шантажист требовал отдать значительную долю кыртовых полей Файфа за эту информацию. Сквайр подозревал, что если преступник смог похитить космоаналитика, прятать его целый год от властей и при этом шантажировать самого главного человека на всем Сарке, он тоже должен быть Великим сквайром. И больше всего подозрений вызывал у него сквайр Стин.

Тут неожиданно Рик вспомнил, что во время разговора с похитителем, тот сообщил размер его владений на Флорине — k квадратных километров. В архиве хранятся данные о площади земель s_i, контролируемых сквайром Стином, в n моментов времени. Файфу известно, что все s_i различны, а так же что до некоторого момента эти площади увеличивались, а потом начали убывать. Более формально, существует такое $1 \leqslant t \leqslant n$, что для любого $1 < i \leqslant t$ $s_{i-1} < s_i$ и для любого $t < j \leqslant n$ $s_{j-1} > s_j$. Чтобы подтвердить свою правоту, Великому сквайру нужно узнать, в какой момент времени площадь владений Стина в точности равнялась k, и он просит вас помочь ему. Вы можете по моменту времени i узнать размер владений сквайра Стина s_i. Небольшая сложность состоит в том, что времени у Файфа мало, а направление запроса в архив происходит достаточно долго. Поэтому у вас есть возможность задать не более 80 таких запросов. Сквайр не сомневается в своем в успехе, и гарантирует вам, что искомое s_i существует.

Протокол взаимодействия

Изначально вам заданы два числа n, k — количество записей о владениях Стина в архиве и значение площади, которое интересует Файфа ($2 \leqslant n \leqslant 2 \cdot 10^5, 0 \leqslant k \leqslant 10^9$). Далее ваша программа может делать запросы вида "? i", в качестве ответа на которые программа жюри будет выводить значения s_i. Все s_i - целые числа, $0 \leqslant s_i \leqslant 10^9$. Записи в архиве нумеруются с 1. Когда ваша программа будет готова дать ответ, она должна вывести "! i", где $s_i = k$, и завершиться. Если ваша программа сделает больше 80 запросов первого типа, решение получит вердикт "Wrong answer". Если программа не завершится после запроса второго типа или ответ на запрос второго типа будет неверным, решение также получит вердикт "Wrong answer".

Пример

стандартный ввод	стандартный вывод
5 3 2 8 10 1 3	? 5 ? 4 ? 3 ? 1 ? 2 ! 2

Замечание

В данном примере записи о владениях выглядели так: $1, 3, 10, 8, 2$.

Задача C. Угадай массив

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

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

Алиса и Боб решили сыграть в игру под названием «Угадай массив». Правила игры очень простые: Алиса загадала массив целых чисел размера n, а Бобу нужно угадать этот массив, сделав не больше n запросов о сумме на отрезке.

За один ход Боб может сделать к Алисе запрос одного из двух типов:

1. «`? l r`» — узнать сумму чисел на отрезке массива с l-го по r-й элемент включительно;
2. «`!`» — сообщить Алисе, что он готов дать ответ. После этого запроса Алиса ожидает от Боба n целых чисел — загаданный массив.

На каждый запрос первого типа, Алиса говорит Бобу сумму чисел на запрошенном отрезке. Но чтобы отгадывать было сложнее, после каждого запроса Алиса делает один отрезок *запрещенным*. В дальнейших запросах Боб не может запрашивать сумму чисел на запрещенных отрезках.

Помогите Бобу угадать загаданный массив, сделав не более n запросов первого типа.

Протокол взаимодействия

В первой строке ввода дано целое число n — размер загаданного массива ($1 \leqslant n \leqslant 10^4$). Гарантируется, что все числа в массиве по модулю не превосходят 10^9.

Далее запускается протокол взаимодействия с программой жюри — интерактором.

Интерактор ожидает от вашей программы запросы двух типов: «`? l r`» или «`!`», где l, r — целые числа — границы отрезка, на котором вы хотите узнать сумму ($1 \leqslant l \leqslant r \leqslant n$). После каждого запроса должен следовать перевод строки. При несоблюдении вашей программой формата запросов, ваше решение может получить произвольный вердикт (отличный от `OK`).

После запроса первого типа необходимо считать со стандартного ввода три целых числа s, l_b и r_b — сумму чисел на запрошенном отрезке s и границы нового *запрещенного* отрезка $[l_b, r_b]$, на котором запрашивать сумму больше нельзя ($|s| \leqslant 10^{18}$; $1 \leqslant l_b \leqslant r_b \leqslant n$).

Запрос второго типа означает, что ваша программа готова дать ответ на задачу. После запроса второго типа необходимо вывести n целых чисел — загаданный массив.

Обратите внимание, ваша программа может сделать не более n запросов первого типа. При превышении данного ограничения, а также при попытке узнать сумму на *запрещенном* ранее отрезке, интерактор выведет «`-1 -1 -1`» и завершится с вердиктом `WA`. Чтобы не получить вердикт `TL` или `IL`, считав из ввода значения «`-1 -1 -1`», ваша программа должна завершить свою работу с нулевым кодом возврата.

Пример

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

Задача D. Поиграем?

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Это **интерактивная** задача.

1804 год. Вице-президент Соединённых Штатов Аарон Бёрр вызывает на дуэль кандидата в губернаторы штата Нью-Йорк Александра Гамильтона за серию оскорбительных памфлетов в свой адрес.

Но Бёрр разумный человек и понимает, что даже если он убьет Гамильтона на дуэли, он потеряет свою репутацию, и его политическая карьера будет закончена. Поэтому противники решили просто сыграть в игру. Для честности они решили сыграть в неё g раз.

Каждую игру Гамильтон загадывает целое положительное число n, а Бёрр пытается его отгадать. Для любого целого положительного x Бёрр может спросить у Гамильтона долю чисел между 1 и n включительно, которые делятся на x. Иными словами, спрашивая про x, он получает значение выражения

$$\frac{\left\lfloor \frac{n}{x} \right\rfloor}{n},$$

причём Гамильтон сообщает ему результат в виде **несократимой** дроби (здесь $\lfloor r \rfloor$ обозначает результат округления вниз вещественного числа r).

Помогите Бёрру найти ответ за некоторое заранее определённое число запросов.

Протокол взаимодействия

При запуске решения на вход подается целое число g — число игр между Гамильтоном и Бёрром ($1 \leqslant g \leqslant 1000$).

Для каждого теста зафиксировано число q ($6 \leqslant q \leqslant 60$) — максимальное количество запросов в одной игре. Гарантируется, что q запросов достаточно, чтобы решить задачу. Эти числа не сообщаются программе участника, но ограничения на эти числа в различных подзадачах приведены в таблице системы оценивания. Если программа участника делает более q запросов программе жюри, на этом тесте она получает в качестве результата тестирования «Wrong answer».

Чтобы сделать запрос, следует вывести строку «`X` t», где t — целое положительное число ($1 \leqslant t \leqslant 10^{18}$), для которого требуется узнать значение выражения

$$\frac{\left\lfloor \frac{n}{t} \right\rfloor}{n}$$

В ответ на каждый запрос программа получает через пробел два числа a и b — числитель и знаменатель этой дроби **после сокращения** — или число -1 в случае, если программа превысила ограничение по числу запросов.

Если программа определила загаданное число, то она должна вывести строку «`N` t», где t — предполагаемый ответ ($1 \leqslant t \leqslant 10^{18}$). Если ответ был правильный, то в ответ программа получает строку «`Correct`», а если неправильный, то она получает строку «`Wrong`».

После этого программа переходит к следующей игре, если они остались, иначе она должна завершиться.

Обратите внимание, в случае считывания числа -1 или строки «`Wrong`» вы **обязательно** должны сразу завершить вашу программу. В противном случае вердикт тестирующей системы может быть некорректным, в частности, вы можете получить вердикт Run-time error или Time limit exceeded!

Гарантируется, что в каждом тесте загаданные числа фиксированы в самом начале игры и не изменяются в зависимости от ваших запросов.

Система оценки

Тесты к этой задаче состоят из девяти групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов **предыдущих** групп.

Группа	Баллы	Дополнительные ограничения
0	0	$q = 60$
1	30	$q = 60$
2	10	$q = 30$
3	4	$q = 20$
4	4	$q = 15$
5	5	$q = 10$
6	5	$q = 9$
7	11	$q = 8$
8	10	$q = 7$
9	21	$q = 6$

Пример

стандартный ввод	стандартный вывод
2 1 2 3 10 1 5 1 5 1 10 1 10 0 1 Correct 1 1 0 1 Correct	 X 2 X 3 X 5 X 4 X 6 X 10 X 11 N 10 X 1 X 2 N 1

Замечание

В первом примере $g = 2$. Приведены примеры запросов, по которым игрок угадывает, что в первой игре загадано число 10, а во второй 1. Эти же числа загаданы в первом тесте в тестирующей системе.

В точности соблюдайте формат выходных данных. После каждого вывода обязательно выводите один перевод строки и сбрасывайте буфер вывода — для этого используйте `flush(output)` на языке Паскаль или Delphi, `fflush(stdout)` или `cout.flush()` в C/C++, `sys.stdout.flush()` на языке Python, `System.out.flush()` на языке Java.

Задача E. Новогодний и прямоугольный

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Это **интерактивная** задача.

На Новый год Дед Мороз подарил Глебу то, о чём он уже давно мечтал — клетчатый квадрат размером $n \times n$. Подарок этот не простой, а с сюрпризом — внутри квадрата Дед Мороз выбрал некоторый непустой прямоугольник, и в каждую клетку этого прямоугольника он положил по мандарину.

Теперь, чтобы получить желанный подарок, Глебу нужно сыграть с Дедом Морозом в очень интересную игру. Глеб должен отгадать, в каком именно прямоугольнике находятся все мандаринки, подаренные Дедом Морозом. Будем считать, что строки и столбцы занумерованы числами от 1 до n снизу вверх и слева направо. Глеб может производить два типа запросов:

- ? $x_1\, y_1\, x_2\, y_2$ ($1 \leqslant x_1 \leqslant x_2 \leqslant n$, $1 \leqslant y_1 \leqslant y_2 \leqslant n$) — в ответ на этот запрос Дед Мороз говорит, сколько мандаринок находится в прямоугольнике, левым нижним углом которого является клетка (x_1, y_1), а правым верхним — клетка (x_2, y_2);
- ! $x_1\, y_1\, x_2\, y_2$ ($1 \leqslant x_1 \leqslant x_2 \leqslant n$, $1 \leqslant y_1 \leqslant y_2 \leqslant n$) — когда Глеб уверен, что он точно знает, где находятся мандаринки, он должен сделать запрос такого вида, чтобы сообщить свой ответ. При этом (x_1, y_1) соответствует предполагаемому расположению левого нижнего угла, а (x_2, y_2) — правого верхнего.

Формат входных данных

При запуске решения на вход вашей программе подается одно число n ($1 \leqslant n \leqslant 2 \cdot 10^9$) — размер квадрата.

Затем на каждый запрос типа "?" вам будет выдаваться количество мандаринок, находящихся в указанном вами прямоугольнике.

Формат выходных данных

Вы должны выводить корректные запросы в формате, описанном выше. Последним должен следовать единственный запрос вида "**!**", после чего ваша программа должна немедленно завершиться. Ваша программа должна произвести не больше q (параметр зависит от номера группы) запросов типа "?". Обратите внимание, что последний запрос, выводящий ответ, не входит в данные q запросов.

В точности соблюдайте формат выходных данных. После вывода каждой строки сбрасывайте буфер вывода — для этого используйте команды `flush(output)` на языке Паскаль или Delphi, `fflush(stdout)` или `cout.flush()` в C/C++, `sys.stdout.flush()` на языке Python, `System.out.flush()` на языке Java.

Примеры

стандартный ввод	стандартный вывод
4 6 6 4	? 1 1 4 4 ? 1 3 4 4 ? 2 3 4 4 ! 1 3 3 4

Замечание

Пример в условии иллюстрирует взаимодействие с проверяющей программой. Для прохождения первого теста не обязательно производить такие же запросы, как в примере.

Тесты к этой задаче состоят из шести групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов **предыдущих** групп.

Группа	Тесты	Баллы	Дополнительные ограничения		Комментарий
			n	q	
0	1 – 1	0	$n = 4$	$q = 1000$	Тест из условия.
1	2 – 12	10	$n \leqslant 10$	$q = 10\,000$	
2	13 – 23	20	$n \leqslant 100$	$q = 10\,000$	
3	24 – 34	20	$n \leqslant 10\,000$	$q = 20\,000$	
4	35 – 45	25	$n \leqslant 2 \cdot 10^9$	$q = 128$	
5	46 – ∞	25	$n \leqslant 2 \cdot 10^9$	$q = 64$	

Задача F. Игра с бинарной строкой

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

Это интерактивная задача.

Алиса и Боб играют в игру. У них есть строка длины n, каждый символ строки — это либо `0`, либо `1`. Игроки ходят по очереди, начинает Алиса. На каждом ходу игрок должен изменить **ровно** один символ строки: `0` меняется на `1`, а `1` — на `0`. После хода игрока должна получиться строка, которая раньше в игре никогда не встречалась (в том числе до всех ходов). Если игрок не может сделать ход, то он проигрывает.

Вам нужно выбрать, за кого (Алису или Боба) вы хотите играть, проверяющая система будет играть за другого игрока. Вы должны выиграть игру за выбранного игрока.

Протокол взаимодействия

В начале вы должны считать n ($1 \leqslant n \leqslant 15$) — длину строки — и строку s длины n — начальное состояние строки. После этого выведите «`Alice`» (если вы хотите играть за Алису) или «`Bob`» (если хотите играть за Боба).

В каждый свой ход вы должны вывести одно число p ($0 \leqslant p \leqslant n$).

- $p = 0$ символизирует, что вы сдаётесь. $1 \leqslant p \leqslant n$ — позиция символа, который вы меняет своим ходом. Позиции в строке нумеруются слева направо от 1 до n.

В каждый ход соперника вы должны считать одно число p ($-1 \leqslant p \leqslant n$).

- $p = 0$ символизирует, что соперник сдаётся. В этом случае вы должны завершить выполнение программы.
- $p = -1$ символизирует, что ваш последний ход привёл к строке, которая ранее встречалась, либо вы совершили недопустимый ход. В этом случае вы также должны завершить выполнение программы, иначе вы можете получить произвольный вердикт. $1 \leqslant p \leqslant n$ — позиция символа, который соперник меняет своим ходом.

Обратите внимание, что если вы выбрали Алису, то вы делаете первый ход, а если вы выбрали Боба, то вы делаете второй ход.

Не забудьте после каждого хода выполнять операцию '`flush`'.

Для сброса буфера вывода (то есть для операции '`flush`') сразу после вывода хода и перевода строки нужно сделать:

- fflush(stdout) в языке C++;
- System.out.flush() в Java;
- stdout.flush() в Python;
- flush(output) в Pascal;
- смотрите документацию для других языков.

В случае, если вы не будете выполнять операцию '`flush`' после каждого хода, либо не будете соблюдать формат взаимодействия с программой жюри, вы можете получить любой вердикт.

Пример

стандартный ввод	стандартный вывод
2 00 1 0	Alice 2 2

Задача G. Погоня в метро

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

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

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

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

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

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

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

Протокол взаимодействия

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

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

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

После каждого запроса необходимо вывести перевод строки и сбросить буфер вывода — для этого используйте команды `flush(output)` на языке Паскаль или Delphi, `fflush(stdout)` или `cout.flush()` в C/C++, `sys.stdout.flush()` на языке Python, `System.out.flush()` на языке Java. В точности соблюдайте формат взаимодействия с системой.

Пример

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

Замечание

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

Задача H. Угадать количество делителей

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

Это интерактивная задача.

Загадано число $1 \leqslant X \leqslant 10^9$. Вам **не нужно** угадывать это число. Вам нужно **определить количество делителей** этого числа, и даже это вам **не нужно делать точно**: ваш ответ будет считаться верным, если его абсолютная погрешность не превышает 7 **или** его относительная погрешность не превышает 0.5. Формально, пусть ваш ответ равен ans, а количество делителей X равно d, тогда ваш ответ будет считаться правильным, если выполнено **хотя бы одно** из следующих двух условий:

- $|ans - d| \leqslant 7$
- $\frac{1}{2} \leqslant \frac{ans}{d} \leqslant 2$

Вы можете не более 22 раз сделать запрос. Запрос состоит из одного числа $1 \leqslant Q \leqslant 10^{18}$. В ответ на запрос вы получите $gcd(X, Q)$ — наибольший общий делитель X и Q.

Число X зафиксировано до всех запросов. Иными словами, **интерактор не является адаптивным**.

Назовём процесс отгадывания количества делителей числа X *игрой*. В рамках одного теста вам нужно будет сыграть T независимых игр, то есть отгадать количество делителей T раз для T независимых чисел X.

Формат входных данных

На первой строке записано одно целое число T ($1 \leqslant T \leqslant 100$) — количество игр.

Протокол взаимодействия

Чтобы сделать запрос, выведите строку вида «`? Q`» ($1 \leqslant Q \leqslant 10^{18}$). После запроса считайте одно число — $gcd(X, Q)$. Вы можете сделать не более 22 таких запросов в рамках одной игры.

Если вы считаете, что знаете количество делителей X с достаточной точностью, выведите ваш ответ в формате «`! ans`». ans должно быть целым числом. Если это последняя игра, то вы должны завершить выполнение программы, иначе вы должны начать следующую игру. Обратите внимание, что интерактор не выводит ничего в ответ на вывод ответа.

После вывода запроса или ответа не забудьте вывести перевод строки и сбросить буфер вывода. Для сброса буфера вывода используйте:

- `fflush(stdout)` или `cout.flush()` в C++;
- `System.out.flush()` в Java;
- `flush(output)` в Pascal;
- `stdout.flush()` в Python;
- смотрите документацию для других языков.

Пример

стандартный ввод	стандартный вывод
2 1 1 1 1024 1048576 4194304	 ? 982306799268821872 ? 230856864650023977 ? 134690134760714371 ! 5 ? 1024 ? 1048576 ? 1073741824 ! 42

Замечание

Почему ограничение на запросы именно 22? Возможно, автор задачи — фанат Тейлор Свифт.

Рассмотрим пример из условия.

В первой игре загадано число $X = 998\,244\,353$. Было бы сложно это угадать, правда? Это число является простым, то есть количество его делителей равно 2. Решение сделало запросы с несколькими случайными числами, и ответы на все запросы оказались равны 1 (удивительно, что ни один из трёх запросов не оказался кратным $998\,244\,353$). Логично предположить, что у загаданного числа не очень много делителей, поэтому решение ответило 5. Почему бы и не 5. Этот ответ будет засчитан, так как $|5 - 2| = 3 \leqslant 7$.

Во второй игре загадано число $X = 4\,194\,304 = 2^{22}$, количество его делителей равно 23. Решение сделало запросы $1024 = 2^{10}$, $1\,048\,576 = 2^{20}$, $1\,073\,741\,824 = 2^{30}$ и получило ответы $1024 = 2^{10}$, $1\,048\,576 = 2^{20}$, $4\,194\,304 = 2^{22}$, соответственно. Затем решение окончательно запуталось и выдало ответ на Главный вопрос жизни, Вселенной и всего такого. Этот ответ будет засчитан, так как $\frac{1}{2} \leqslant \frac{42}{23} \leqslant 2$.

Задача I. Игры уголовников

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Случилось ужасное! Товарищ майор узнал, что вы репостнули мем про Иисуса, написали комментарий, оскорбляющий власть, лайкнули очередной видосик Навального, да и ещё и сходили на нанесанкционированный митинг! Теперь вы официально экстремист, а за это вам грозит тюремный срок на 100 лет!

В тюрьме вы сразу повстречали заядлого уголовника Антона, который предложил вам сыграть в особую тюремную игру уголовников. Игра заключается в следующем: есть 2 игрока, один — чётный, а другой — нечётный. В каждом раунде игры оба игрока показывают один или два пальца. Если a — сумма показанных пальцев, то если a чётно, то нечётный игрок платит чётному a монет, а если a нечётно, то чётный игрок платит нечётному a монет. Условия одинаковы, поэтому Антон предложили вам самим выбрать, чётным вы будете или нечётным. После этого каждый день своего десятилетнего заключения (всего $365 * 100 = 36500$ дней) вы сыграете с Антоном по партии. Ваша цель — не остаться в минусе и в конце своего заключения не потерять свои деньги после всех 36500 партий. Вам лениво каждый раз приходить к Антону и играть с ним в это игру, поэтому вы решили написать программу, играющую за вас.

Протокол взаимодействия

Это интерактивная задача.

В первой строке выведите 0, если вы хотите играть за чётного игрока, или 1, если вы хотите играть за нечётного.

После этого следующие 36500 ходов пройдут следующим образом. Выведите одно число 1 или 2 — количество пальцев, которое вы показываете в этом раунде. В ответ на это программа жюри, имитирующая Антона, выведет вам одно число 1 или 2 — количество пальцев, которое показывает Антон. Гарантируется, что в i-й ход выведенное интерактором число никак не зависит от того, какое число вы вывели в i-й ход, т.е. программа Антон играет честно.

После 36500 ходов программа жюри выведет ваш баланс — разницу полученных и отданных вами монет. В случае, если баланс неотрицательный, ваша программа будет считаться верной.

Все строки завершайте символом перевода строки и сбросом буфера ввода. Все числа программы жюри выводятся в новой строке. Для удобства в примере показана игра, состоящая из 4 партий.

Пример

стандартный ввод	стандартный вывод
	0
	1
1	
	1
2	
	2
1	
	2
2	
0	

Задача J. Дом для червячка

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Это интерактивная задача.

Вы ищете место в земле, куда вы бы могли поместить свое любимое домашнее животное — червячка Максимуса! В качестве зоны поисков вы выбрали клочок земли размера $N \times M \times K$. Выбранный клочок земли поделен на $N \cdot M \cdot K$ единичных кубических клеток размера $1 \times 1 \times 1$. Каждая клетка имеет свои координаты (x, y, z), где $1 \leqslant x \leqslant N$, $1 \leqslant y \leqslant M$ и $1 \leqslant z \leqslant K$. Также для каждой клетки при помощи специального сканера может быть определена ее влажность $h(x, y, z)$, которая является целым числом от 1 до 10^9. При желании вы можете измерить влажность любой клетки выбранного клочка земли.

Максимус очень любит влажную почву, поэтому вы хотите выбрать для него такую клетку земли, влажность которой не меньше, чем влажность всех шести соседних клеток. В противном случае червячок может не согласиться с вашим выбором и уползти в соседние клетки, после чего найти его будет крайне сложно. Формально, от вас требуется найти клетку (x, y, z), которая является *локальным максимумом* влажности, то есть для нее должно выполняться условие:

$h(x, y, z) \geqslant \max(h(x-1, y, z), h(x+1, y, z), h(x, y-1, z), h(x, y+1, z), h(x, y, z-1), h(x, y, z+1))$.

Для удобства будем считать влажность клеток, находящихся за пределами выбранного участка, равной нулю.

Количество клеток в выбранном клочке земли может быть достаточно большим, и, разумеется, вы не хотите потратить слишком много времени, чтобы измерить влажность всех клеток. Поэтому перед вами встала непростая задача — измерить влажность не слишком большого количества клеток и найти клетку (x, y, z), обладающую нужными свойствами.

Протокол взаимодействия

В начале взаимодействия с программой жюри вы должны считать четыре целых числа N, M, K и Q — размеры клочка земли, а также максимальное количество измерений влажности, которое вам разрешено сделать.

После этого вы можете не более, чем Q раз выяснить влажность почвы в некоторой клетке, выведя запрос вида: «`? x y z`» ($1 \leqslant x \leqslant N$, $1 \leqslant y \leqslant M$, $1 \leqslant z \leqslant K$). После вывода запроса вы должны считать ответ программы жюри — одно целое число в диапазоне от 1 до 10^9 — влажность почвы в клетке (x, y, z). В случае, если вы превысили количество допустимых запросов, в качестве ответа вы получите число -1. В этом случае вы должны немедленно завершить выполнение программы.

В случае, если вы нашли клетку (x, y, z), являющуюся подходящей для размещения в ней червячка, вы должны вывести запрос вида: «`! x y z`» ($1 \leqslant x \leqslant N$, $1 \leqslant y \leqslant M$, $1 \leqslant z \leqslant K$) и немедленно завершить выполнение программы.

После каждого запроса, в том числе после последнего, вы должны выполнять операцию `flush`. Примеры выполнения данной операции для разных языков программирования приведены ниже:

- Java: `System.out.println()` выполняет операцию `flush` автоматически.
- Python: `print()` выполняет операцию `flush` автоматически.
- C++: `cout << endl` выполняет операцию `flush` автоматически. В случае использования `printf`, выполняйте операцию `fflush(stdout)`.
- Pascal: `Flush(Output)`.

В случае несоблюдения описанного протокола ваше решение может получить любой вердикт.

Гарантируется, что программа жюри **не является адаптивной**. Это значит, что каждый тест зафиксирован до запуска решения и не может меняться в процессе общения решения и программы жюри.

Система оценки

Баллы за каждую группу выдаются только в случае, если все тесты группы успешно пройдены. Ограничения на N, M, K и Q записаны в таблице ниже. Группы оцениваются независимо друг от друга.

Группа	Баллы	N	M	K	Q
0	0	Тесты из условия			
1	10	$N = 1\,000\,000$	$M = 1$	$K = 1$	$Q = 10\,000$
2	22	$N = 1\,000\,000$	$M = 1$	$K = 1$	$Q = 35$
3	12	$N = 200$	$M = 200$	$K = 1$	$Q = 4\,000$
4	19	$N = 1\,000$	$M = 1\,000$	$K = 1$	$Q = 3\,500$
5	14	$N = 100$	$M = 100$	$K = 100$	$Q = 100\,000$
6	23	$N = 500$	$M = 500$	$K = 500$	$Q = 150\,000$

Пример

стандартный ввод	стандартный вывод
3 1 1 3 13 14 10	 ? 3 1 1 ? 2 1 1 ? 1 1 1 ! 2 1 1

Задача K. Conv19

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

Это интерактивная задача.

В свете недавних событий было решено отменить чемпионат мира по шахматам. Гроссмейстеры, сдав обратно авиабилеты, начали думать, чем бы занять свое освободившееся время. Они наткнулись на шахматную онлайн игру «Conv19». В этой игре есть шахматное поле $N \times N$ и всего одна фигура — клетка вируса. Эта клетка бьет некоторые фигуры следующим образом: перед началом игры фиксируется нечетное число $M \geqslant 3$; берется черно-белая шахматная раскраска квадрата 5×5, где центр является клеткой черного цвета; после этого каждая клетка такой раскраски меняется на квадрат из $M \times M$ клеток. Если вирус находился в клетке (x, y), то множество клеток, которые находились под его ударом, определялось так: центр получившегося квадрата размером $5M \times 5M$ прикладывался к точке (x, y), и все черные клетки считались клетками под ударом. При этом считается, что вирус мог находиться только в тех клетках поля, куда можно было приложить весь квадрат (то есть, сделать это так, чтобы он не вылез за границы поля).

Процесс игры выглядит так. Один игрок кладет вирус в произвольную клетку поля. После этого ходы делает только второй игрок. За ход он может узнать про произвольную клетку — бьет ее вирус или нет. Цель второго игрока — отгадать, где находится вирус, за не более, чем 300 ходов.

Костя решил сыграть в эту игру с Ваней. Поскольку на большом поле игра может затянуться надолго, Костя решил дать Ване подсказку, и указал ему на точку, которая бьется вирусом. Ваня решил, что готовить контест важнее, чем играть в какие-то игры, поэтому попросил вас написать программу, которая выиграет за него. Обратите внимание, что число M Костя выберет сам, и не скажет его Ване.

Протокол взаимодействия

В начале выполнения вашей программе на вход подается число N ($15 \leqslant N \leqslant 2\,000\,000\,000$), а также координаты клетки, которую бьет вирус X, Y ($1 \leqslant X \leqslant N$, $1 \leqslant Y \leqslant N$).

После этого вы можете выводить запросы «examine x y», где $1 \leqslant x, y \leqslant N$, чтобы узнать, бьет ли вирус клетку (x, y). Интерактор ответит вам «true» или «false», соответствующие ответу на этот вопрос.

В тот момент, когда вы посчитали, что знаете клетку с вирусом, выведите ответ в формате «solution x y».

Пример

стандартный ввод	стандартный вывод
20 4 9 false true false true true false true false true	examine 2 9 examine 3 9 examine 6 9 examine 5 9 examine 4 3 examine 2 3 examine 3 3 examine 3 1 examine 3 2 solution 10 9

Замечание

Соблюдайте формат ввода-вывода, не забывайте очищать за собой буфер, и не болейте.

Задача L. Лошадь и Ежик

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

Это интерактивная задача.

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

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

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

Протокол взаимодействия

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

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

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

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

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

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

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

Пример

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

Замечание

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

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

Задача M. Странный прибор

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

Это интерактивная задача.

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

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

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

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

Протокол взаимодействия

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

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

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

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

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

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

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

Чтобы сбросить буфер вывода вы можете использовать:

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

Пример

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

Замечание

Таблица попарных расстояний между вершинами выглядит так:

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

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

Задача N. Петербург?

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

– Это что за остановка –
Бологое иль Поповка? –
А с платформы говорят:
– Это город Ленинград.

«Вот какой рассеянный», Самуил Маршак

Пытаясь спастись от мира спортивного программирования, Алина сбежала на вокзал и уехала прочь на ночной электричке. Минуты медленно уплывали в даль, и уставшую девочку клонило в сон. Ей снился город-сказка, где не надо программировать, а можно гулять, мечтать и наслаждаться жизнью. Внезапно дождь из **интерактивных** задач разрушил эту идиллию.

Проснувшись и открыв окно, Алина задалась вопросом весьма философского свойства: «Где я?». С перрона потерявшейся девочке сообщили, что этот город, не похожий ни на что вокруг, представляет собой неориентированный граф на n вершинах и m ребрах. Сей невероятный факт, однако, нисколько не удивил Алину. Она давно мечтала побывать в одном таком городе — Петербурге. Его уникальной отличительной особенностью является то, что хотя бы **половина** его ребер — мосты (определение дано в конце условия). Так как никакие другие города Алине не интересны, она решила ограничиться расспросом находящихся на платформе эрудированных путешественников. Любой из их них может по данной вершине v сообщить любое ещё не названное ребро, исходящее из нее, или же заявить об отсутствии таковых.

Алина неуверена в своих силах, поэтому попросила вас помочь ей определить, попала ли она в Петербург. Так как её поезд скоро продолжит свой путь, задать больше $3n$ вопросов не получится.

Обратите внимание, что в графе **могут** присутствовать петли и кратные ребра.

Протокол взаимодействия

В первой строке стандартного потока ввода даны два целых числа n и m ($1 \leqslant n, m \leqslant 100\,000$) — число вершин и ребер в графе соответственно.

Для того, чтобы узнать очередное ребро, исходящее из u-й вершины ($1 \leqslant u \leqslant n$), нужно вывести «`?` u». После этого ваша программа на вход получит целое число v ($-2 \leqslant v \leqslant -1$ или $1 \leqslant v \leqslant n$) — $v = a + b - u$, если существует ребро ab, которое инцидентно вершине u и **ещё не было названо**, -1, если такого ребра не существует и -2, если вы превысили допустимое число запросов. В последнем случае ваша программа должна немедленно завершиться, в ином случае жюри не гарантирует корректность полученного вами вердикта.

Вам разрешается задать не более $3n$ вопросов.

Чтобы сообщить, что ответ найден, требуется вывести «`! Yes`» или «`! No`», в зависимости от того, является ли загаданный граф Петербургом. В случае положительного ответа выведите $\lceil \frac{m}{2} \rceil$ строк, по два целых числа u_i и v_i в каждой ($1 \leqslant u_i, v_i \leqslant n$), обозначающих, что ребро (u_i, v_i) является мостом. Любое ребро в приведенном списке должно встречаться не более одного раза (кратные ребра считаются различными).

Запрос на вывод ответа не входит в ограничение на $3n$ запросов.

Не забывайте сбрасывать буфер после каждого запроса. Например, на языке C++ надо использовать функцию `fflush(stdout)` или вызов `cout.flush()`, на Java вызов `System.out.flush()`, на Pascal `flush(output)` и `stdout.flush()` для языка Python.

Примеры

стандартный ввод	стандартный вывод
3 3 2 2 -1 3 -1 -1	 ? 3 ? 1 ? 2 ? 1 ? 1 ? 3 ! No
4 4 2 3 2 -1 4 -1 -1 -1	 ? 1 ? 2 ? 3 ? 1 ? 3 ? 3 ? 2 ? 4 ! Yes 1 2 3 4

Замечание

В условии в примере взаимодействия вводимые и выводимые данные расположены для удобства восприятия в хронологическом порядке, при реальном взаимодействии никакие «лишние» переводы строк возникать не должны.

Ввод-вывод в примерах демонстрирует пример взаимодействия вашей программы с проверяющей системой.

В первом примере был загадан граф на трех вершинах с ребрами $(1, 2)$, $(2, 3)$ и $(3, 1)$.

Во втором примере была загадан граф на четырех вершинах с ребрами $(1, 2)$, $(2, 3)$, $(3, 4)$ и $(2, 3)$.

Ребро, соединяющее вершины u и v, называется мостом, если после его удаления между вершинами u и v не существует пути.

Задача O. Alien Invasion Online

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 megabytes

This is an interactive problem.

Recently, aliens from planet *Nibiru*, also known as *Not-Taking-It*, have decided to invade the Earth. As the chief of the Nibiru Special Forces headquarters, you are planning to land the first group of spaceships in Byteland, the most peaceful country of the Earth.

You know that the territory of Byteland has a shape of a polygon with n vertices on a plane (yes, Byteland is not very large, so we can view the surface around Byteland as part of the plane). Each vertex of this polygon represents a town, and each side represents a part of the border of Byteland. Of course, the border cannot intersect itself, and two sides of the polygon can have a common point only if they are neighboring (in this case, there is exactly one common point).

Your task is to find the exact area of Byteland in order to calculate the number of spaceships to be located there. However, you do not know the exact shape of Byteland, since it would take too much time to discover. Fortunately, the Nibiru Intelligence Agency has sent spies in each of the n towns of Byteland. Each spy has a huge signal lamp. You also know that the spies are numbered 1 through n such that the spies with neighboring numbers (in other words, numbers that differ either by 1 or by $n-1$) are located in towns that are directly connected by a part of the border.

You can perform the following *special operation*. At first, you tell each spy to switch his lamp either on or off. Then, you send an astronaut to the Earth, she sees all lighted lamps and tells you the least area of a convex set containing all these lamps (in other words, the area of the convex hull of these lamps). As it is difficult to keep orientation in space, the astronaut can tell nothing else about positions of the towns.

Note that if you perform a *special operation* more than $\frac{n(n-1)}{2}$ times, the Bytelandian police will notice something strange and the invasion will fail. Find the exact area of Byteland without making too much noise!

Протокол взаимодействия

At first, you get one integer n ($3 \leqslant n \leqslant 200$), the number of vertices of the polygon representing Byteland.

Then you make at most $\frac{n(n-1)}{2}$ *special operations*. A query describing the special operation should have the form of two lines:

`?` k

a_1 a_2 ... a_k

where k ($3 \leqslant k \leqslant n$) is the number of spies that have to switch on their lamps and $a_1, \ldots, a_n$ ($1 \leqslant a_i \leqslant n$, $a_i \neq a_j$ for $i \neq j$) are the indices of towns where these spies are located. All spies in all other towns will switch off their lamps.

After making a query, you should read one line with a single non-negative number from the input: the area of the convex hull built on lamps that are lit up (in square meters).

When you think you know the area x of Byteland, print

`!` x

on a single line. Printing the answer does not count as a special operation. The answer should be accurate, neither any errors nor leading zeros are allowed. After that, your program should terminate gracefully.

Remember to put a newline character and flush the output buffer after making a query or printing an answer.

It is guaranteed that the polygon does not change during the program execution, and the area of the polygon is strictly greater than zero. Also, it is guaranteed that there exists a Cartesian coordinate system on a plane containing Byteland such that both coordinates of all vertices of the polygon (in meters) are integers in the range $[-10^9, 10^9]$.

Пример

standard input	standard output
3 0.5 	 ? 3 1 2 3 ! 0.5

Замечание

For the sample test, the Bytelandian towns are located in points $(0, 0)$, $(0, 1)$ and $(1, 0)$ for some Cartesian coordinate system.

Note that blank lines in sample input and output are printed only for clarity and do not exist in reality.