Задача A. Поиск буквы «a»

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
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`».

Задача B. Случайные тоннели

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
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 C. Поиск уникального элемента

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
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	Загаданный массив
? 1 ? 5 ? 3 ! 2	5 1 3 2	1 1 2 3 3

Задача D. Джек и Джилл

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
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	>\n>\n*...и так далее...*\n>\n=
1000 999 *...и так далее...* 902 901	<\n<\n*...и так далее...*\n<\n<

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 E. Детская игра с роботом

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

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

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

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

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

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

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

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

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

Протокол взаимодействия

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

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

После вывода десяти команд решение должно корректно завершить свою работу.

Пример

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

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

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

Замечание

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

Задача F. Круговой поиск

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

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

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

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

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

Протокол взаимодействия

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

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

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

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

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

Пример

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

Задача G. Русская рулетка

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

«Русская рулетка» — это игра, в которой участвуют n человек, пронумерованных числами от 1 до n. Изначально все n человек встают в какие-то различные точки плоскости, отличные от начала координат, и каждый задумывает какое-то число. В точке $(0, 0)$ находится рулетка, в начале игры направленная под углом f градусов к оси Ox (угол считается в направлении против часовой стрелки, если число градусов положительно, и по часовой стрелке, если оно отрицательно). Один ход в игре состоит в следующем:

- Найдём человека, расстояние от которого до луча из начала координат в направлении, куда показывает рулетка, минимально; в случае равенства расстояний выбирается человек с меньшим номером.
- Этот человек выходит из игры, а рулетка поворачивается на число градусов, соответствующее числу, загаданному им.

Так продолжается до тех пор, пока в игре не останется всего лишь один человек. Он и считается выигравшим.

Иосиф опоздал к началу расстановки, и в качестве штрафа ему присвоили номер 0. Это означает, что в ситуации, когда расстояние до луча минимально для него и для какого-то другого человека, для выбывания всегда будет выбран Иосиф. Тем не менее, у него ещё может быть шанс выиграть...

Помогите Иосифу встать в такую точку плоскости, чтобы выиграть в этой игре, или выясните, что это невозможно.

Формат входных данных

В первой строке записано два целых числа n и f через пробел ($2 \leqslant n \leqslant 1000$, $-180 \leqslant f < 180$). Следующие n строк содержат по три целых числа x_i y_i t_i каждая через пробел — координаты i-го человека и число, загаданное им ($-1000 \leqslant x_i, y_i, t_i \leqslant 1000$). Никакой человек не стоит в начале координат или в точке, занятой другим человеком.

Формат выходных данных

Если у Иосифа не получится выиграть, выведите строку «`NO SOLUTION`». Иначе выведите два целых числа x_0 и y_0 через пробел — координаты точки, в которую Иосифу надо встать, чтобы выиграть. Координаты не должны превышать по модулю 10 000; кроме того, нельзя вставать в начало координат и в точку, в которой уже кто-то стоит. Если существует несколько решений, можно вывести любое из них.

Пример

стандартный ввод	*стандартный вывод*
4 90 0 1 90 1 0 90 0 -1 90 -1 0 90	2 2

Задача H. Разбиение

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Рассмотрим клетчатую плоскость. Если выбрать множество клеток на плоскости, оно задаёт граф следующим образом: клетки являются вершинами графа, и между двумя клетками есть ребро в графе, если эти клетки имеют общую сторону.

Даны n клеток на плоскости. Гарантируется, что граф, задаваемый этими клетками, связен. Необходимо удалить не более $\left\lceil \frac{3}{2}\sqrt{n} \right\rceil$ клеток так, чтобы в графе, задаваемом оставшимися клетками, каждая компонента связности состояла не более чем из $\left\lceil \frac{n}{2} \right\rceil$ клеток.

Формат входных данных

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

В каждой из последующих n строк записано по два числа x и y — координаты клеток ($-10^9 \leqslant x, y \leqslant 10^9$). Гарантируется, что все n заданных клеток различны.

Формат выходных данных

В первой строке выведите целое число m — количество клеток, которые необходимо удалить. Помните, что это количество *не обязательно* минимизировать, однако оно должно быть не больше $\left\lceil \frac{3}{2}\sqrt{n} \right\rceil$.

В следующих m строках выведите координаты клеток, которые необходимо удалить, в формате, аналогичном формату входных данных. Все выведенные клетки должны быть различны и должны встречаться во входных данных. Клетки можно выводить в любом порядке. Если возможных ответов несколько, можно вывести любой из них.

Примеры

стандартный ввод	*стандартный вывод*
16 1 1 1 2 1 3 1 4 2 1 2 2 2 3 2 4 3 1 3 2 3 3 3 4 4 1 4 2 4 3 4 4	4 1 1 2 2 3 3 4 4

стандартный ввод	*стандартный вывод*
14 1 1 1 2 1 7 1 8 2 1 2 2 2 3 2 4 2 5 2 6 2 7 2 8 3 4 3 5	2 2 3 2 6

Задача I. MST случайных точек

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Даны n различных точек на плоскости. Координаты точек — целые числа от 0 до $30\,000$ включительно. Точки выбраны *случайно* в следующем смысле: рассмотрим все возможные наборы из n различных точек на плоскости с заданными ограничениями на координаты и выберем из них случайно и равновероятно один набор.

Вы можете провести отрезок между любыми двумя заданными точками. Длина отрезка между точками с координатами (x_1, y_1) и (x_2, y_2) равна $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Будем говорить, что точки a и b *связаны*, если они соединены отрезком, или же существует точка d, которая связана и с a, и с b. Ваша задача — провести отрезки минимальной суммарной длины так, чтобы все точки были связаны.

Формат входных данных

В первой строке ввода задано целое число n ($2 \leqslant n \leqslant 50\,000$). Следующие n строк содержат координаты точек. Гарантируется, что все точки различны. Кроме того, во всех тестах, кроме примера, гарантируется, что точки выбраны случайно, как описано в условии.

Формат выходных данных

В первой строке выведите вещественное число w — суммарную длину отрезков. В следующих $(n - 1)$ строках выведите отрезки, по одному на строке. Каждый отрезок следует выводить как два числа от 1 до n, обозначающие номера точек, являющихся концами этого отрезка.

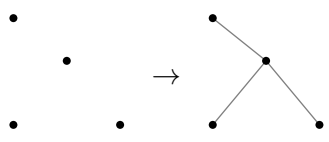
Пусть на самом деле суммарная длина выведенных вами отрезков равна w^*, а суммарная длина отрезков в оптимальном ответе равна w_{opt}. Тогда ваш ответ будет считаться верным, если

$$\max\left(\left|\frac{w}{w^*} - 1\right|, \left|\frac{w^*}{w_{\text{opt}}} - 1\right|\right) < 10^{-12}.$$

Пример

стандартный ввод	*стандартный вывод*
4 0 10 5 6 10 0 0 0	22.02362358924615 1 2 2 3 4 2

Иллюстрация



Задача J. Новая коллекция

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

В коллекционной карточной онлайн-игре CCG вышло новое дополнение. Игроки могут покупать случайные карты дополнения по одной: каждая купленная карта равновероятно и независимо от других покупок окажется одной из карт нового дополнения.

Пётр — один из основных авторов CCG Scene — сайта игроков, куда попадают известные данные обо всех картах CCG, а также их достоинства, недостатки, стратегии использования и многое другое.

Один из важных параметров дополнения — сколько в нём всего карт. Чтобы поддержать свою репутацию в сообществе, Пётр хочет как можно скорее выложить эту информацию. Известно, что в каждом из предыдущих дополнений количество карт было целой степенью числа 10: от десяти до десяти миллионов включительно. Пётр предположил, что это правило соблюдается и в новом дополнении, и что количество карт с одинаковой вероятностью может оказаться каждой из возможных степеней десятки.

Ресурсы Петра позволяют ему купить до 10 000 карт. У каждой карты CCG есть идентификатор, но известно только, что эти идентификаторы одинаковые у совпадающих карт и разные у различных. Пётр может покупать карты по одной, и после каждой покупки узнаёт идентификатор только что купленной карты. Кроме того, в любой момент Пётр может прекратить покупки и написать на CCG Scene количество карт в новом дополнении.

Напишите программу, которая поможет Петру найти правильный ответ.

Протокол взаимодействия

Решение должно выводить каждое действие в стандартный поток вывода на отдельной строке. Действие имеет либо вид «`+`», означающий покупку очередной карты, либо вид «`=` n», означающий вывод ответа.

После каждой покупки в стандартный поток ввода поступает очередная строка — идентификатор только что купленной карты. Идентификатор состоит ровно из двенадцати шестнадцатеричных цифр (`0`–`9` и `A`–`F`), возможно, с ведущими нулями.

После вывода ответа решение должно сразу корректно завершить работу.

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.

В каждом тесте к этой задаче зафиксировано количество карт n — целая степень числа 10 от 10^1 до 10^7 включительно. Гарантируется, что каждая купленная карта равновероятно и независимо от других покупок окажется одной из n возможных карт. Тем не менее, в каждом тесте заранее зафиксировано, моделируя это предположение, какие идентификаторы будут иметь первые 10 000 купленных карт.

Пример

действия участника	ответы проверяющей программы
+	13A17DA944F1
+	0BE186EC4732
+	1591FAA834F2
+	13A17DA944F1
+	13A17DA944F1
+	0E703FE27659
+	1A16F6A9EBB6
+	1591FAA834F2
+	0E703FE27659
+	07B569E7D7A4
+	13A17DA944F1
= 10	

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. Количество карт в дополнении в этом тесте равно 10, а их идентификаторы таковы:
07B569E7D7A4,
0A2001E19A1F,
0BE186EC4732,
0E703FE27659,
121657A5CA39,
13786BA38233,
13A17DA944F1,
1591FAA834F2,
189A2AAE360C,
1A16F6A9EBB6.

Решение сначала покупает одну за другой одиннадцать карт, при этом три карты встретились больше одного раза, а четыре карты вообще не встретились. После этого решение считает, что полученной информации достаточно, чтобы установить, что карт всего 10, и выводит ответ.

Задача K. Многочлен в чёрном ящике

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
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 L. Поиск маленьких чисел

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
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 *(waiting for output)* < *(waiting for output)* > *(waiting for output)* <	*(reading input)* ? 1 2 *(reading input)* ? 3 1 *(reading input)* ? 2 3 *(reading input)* ! 1 2 3 *(terminating)*

Пояснение к примеру

В примере загадана перестановка 1 2 3.

Задача M. Передача по частям

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

Это интерактивная задача.

Разведчица Саша хочет передать в штаб важное сообщение, скрытое в строке из маленьких букв английского алфавита. Сегодня используется следующий алгоритм шифрования: сообщение — это самая часто встречающаяся подстрока переданной строки, а если таких подстрок несколько — лексикографически максимальная из них.

Поскольку передача — дело опасное, Саша собирается разбить строку на k частей и передавать их по порядку — передавать очередную часть, как только представится возможность. Между передачей двух соседних частей может случиться всякое, и нельзя исключать, что после любой части дальнейшая передача будет невозможна. Поэтому Саша хочет проверить, что в таком случае сообщение не получится каким-нибудь катастрофическим.

По заданным k частям в порядке их следования в строке выведите k сообщений: что получится при расшифровке, если передача прекратится после первой, второй, ..., k-й части.

Строка s лексикографически больше строки t, если либо t — собственный префикс s, либо есть позиция, в которой s и t не совпадают, и для самой левой из таких позиций i верно $s_i > t_i$.

Протокол взаимодействия

Ваша программа должна общаться с программой жюри через стандартный поток ввода и стандартный поток вывода.

Сначала на вход поступает число k на отдельной строке ($1 \leqslant k \leqslant 10\,000$). Далее на вход последовательно поступают k частей строки. Каждая часть задана на отдельной строке, состоит из маленьких букв английского алфавита и имеет длину от одной до миллиона букв включительно. Кроме того, гарантируется, что суммарная длина строки — от одной до миллиона букв включительно. Каждая часть, кроме первой, поступает на вход, как только получен ответ, связанный с предыдущей частью.

После получения каждой из k частей сразу выведите строку: каким будет расшифрованное сообщение, если передача в этот момент оборвётся. Строка должна завершаться переводом строки.

Чтобы предотвратить буферизацию вывода, после каждого выведенного сообщения 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.

После вывода k ответов ваша программа должна сразу корректно завершить работу.

Пример

стандартный ввод	*стандартный вывод*
4 a c ba cb	*(reading...)* a c a cb

Задача N. Угадайте две строки

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

Жюри загадало две секретные двоичные строки длины N и обозначило их как s и t так, что строка s лексикографически не больше строки t. Ваша задача — угадать эти строки. Для этого вы можете попросить жюри сгенерировать не более Q строк. Каждую такую строку r жюри сгенерирует следующим образом:

1. начнёт с присваивания $r = s$ или $r = t$, выбрав одну из них случайно с одинаковой вероятностью,
2. случайно выберет K **различных** позиций в строке r таким образом, что каждое множество из K позиций имеет одинаковую вероятность быть выбранным,
3. поменяет значения цифр на выбранных позициях в строке r: все нули заменит на единицы, а все единицы — на нули,
4. выдаст вам получившуюся изменённую строку r.

Заметьте, что s и t не меняются при генерации строки r.

Ваша задача — правильно угадать строки s и t.

Протокол взаимодействия

Изначально вам даётся строка, в которой записаны три числа N, K и Q ($N = 100$, $K = 15$, $Q = 100$) — длина строк s и t, количество позиций для изменения при генерации и максимальное количество строк, которые получится сгенерировать.

Чтобы попросить сгенерировать следующую строку, выведите в отдельной строке один символ «`?`». После этого вам будет выдана строка из N двоичных цифр: очередная сгенерированная строка r. Можно сделать не более Q таких запросов.

Когда вы готовы угадать обе секретные строки, выведите «! s t», где s и t — две строки из N двоичных цифр каждая. После этого корректно завершите работу решения.

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 1 42 1010 1110 0110 0010 0000 0100 0011 0111	 ? ? ? ? ? ? ? ? ! 0010 0110

Пояснение к примеру

Этот пример не подходит под ограничения, и приведён только для пояснения формата взаимодействия. Все тесты в проверяющей системе будут удовлетворять всем ограничениям из условия.

Задача O. Умножение

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