## Задача A. Абстрактное покрытие круга

| | |
|---|---|
| Имя входного файла: | *стандартный ввод* |
| Имя выходного файла: | *стандартный вывод* |
| Ограничение по времени: | 60 секунд |
| Ограничение по памяти: | 512 мебибайт |

*Ограничение по времени жёстче обычного.*

На круге отмечено $n$ точек. Точки пронумерованы по часовой стрелке целыми числами от 0 до $n-1$ включительно. Назовём *циклическим отрезком* длины $\ell$ ($1 \leqslant \ell \leqslant n$) с началом $i$ ($0 \leqslant i \leqslant n-1$) набор из $\ell$ последовательных (по часовой стрелке) точек, начиная с $i$-й (другими словами, набор состоит из точек с номерами $i, (i+1) \bmod n, (i+2) \bmod n, \ldots, (i+\ell-1) \bmod n$). Циклические отрезки длины $n$ с началами в $0, 1, \ldots, n-1$ считаются попарно различными, несмотря на то, что они состоят из одного и того же множества точек.

Каждому циклическому отрезку сопоставлена целая стоимость $c_{i,\ell}$. Для каждого $k$ от 1 до $n$ найдите следующую величину: минимальную возможную суммарную стоимость такого набора из ровно $k$ циклических отрезков, что каждая из $n$ отмеченных точек содержится в ровно одном отрезке набора. Стоимость набора отрезков — просто сумма стоимостей отрезков набора.

Заметьте, что на значения $c_{i,\ell}$ не накладывается **никаких** условий, кроме того, что они — относительно небольшие положительные целые числа. Другими словами, любой массив из $n \times n$ целых чисел от 1 до $10^6$ является корректным тестом для этой задачи.

### Формат входных данных

В первой строке дано целое число $n$ ($1 \leqslant n \leqslant 850$) — количество точек на круге. В $(i+1)$-й ($0 \leqslant i \leqslant n-1$) из оставшихся $n$ строк даны $n$ целых чисел $c_{i,1}, c_{i,2}, \ldots, c_{i,n}$ ($1 \leqslant c_{i,\ell} \leqslant 10^6$ для $1 \leqslant \ell \leqslant n$) через пробел.

### Формат выходных данных

Выведите $n$ целых чисел через пробел: $k$-м из них должна быть минимальная суммарная стоимость $k$ циклических отрезков, покрывающих все точки ровно по одному разу.

### Примеры

| *стандартный ввод* | *стандартный вывод* |
|---|---|
| 3<br>10 12 23<br>7 4 11<br>8 5 3 | 3 12 25 |
| 1<br>15 | 15 |

## Задача B. 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<br><br>7<br><br>34<br><br>98 | ask 1<br><br>ask 3<br><br>ask 6<br><br>ask 10<br><br>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 C. Заклинания

| | |
|---|---|
| Имя входного файла: | *стандартный ввод* |
| Имя выходного файла: | *стандартный вывод* |
| Ограничение по времени: | 10 секунд |
| Ограничение по памяти: | 512 мебибайт |

Вам дана строка $S$. Ваша задача — найти в ней длину наибольшего *заклинания* — подстроки вида $\omega\omega^R\omega\omega^R$, где $\omega^R$ означает строку $\omega$, записанную задом наперёд. Например, в строке «abrahellehhelleh» есть заклинание «hellehhelleh» длины 12, а в строке «rachelhellabracadabra» заклинаний нет вообще. Обратите внимание, что ответ на задачу всегда кратен 4.

### Формат входных данных

В первой строке записано одно число $Z \leqslant 40$ — количество тестовых случаев. Каждая из последующих $Z$ строк содержит по одной строке $S$ ($|S| \leqslant 3{\cdot}10^5$), состоящей из строчных или заглавных букв английского алфавита.

### Формат выходных данных

Выведите $Z$ чисел, каждое на отдельной строке: ответы на задачу для каждого тестового случая.

### Пример

| *стандартный ввод* | *стандартный вывод* |
|---|---|
| 2<br>abrahellehhelleh<br>rachelhellabracadabra | 12<br>0 |

## Задача D. Удвоение прямоугольников

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

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

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

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

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

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

### Формат входных данных

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

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

### Формат выходных данных

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

Выводите ответ без кавычек.

### Примеры

| *стандартный ввод* | *стандартный вывод* |
|---|---|
| 4<br>1 4 | letoucan |
| 4<br>2 3 | letoucan |
| 4<br>1 3 | cdkrot |

### Пояснения к примерам

Промоделируем первый пример.

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

Тогда состояния последовательно изменяются следующим образом:

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

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

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

## Задача E. 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 F. 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 | *(reading input)* |
| *(waiting for output)* | ? 1 2 |
| < | *(reading input)* |
| *(waiting for output)* | ? 3 1 |
| > | *(reading input)* |
| *(waiting for output)* | ? 2 3 |
| < | *(reading input)* |
| | ! 1 2 3 |
| | *(terminating)* |

### Пояснение к примеру

В примере загадана перестановка `1 2 3`.

## Задача G. 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<br><br>9 6<br> | <br>1 2 3 4<br><br>3 |

## Задача H. Партии

| | |
|---|---|
| Имя входного файла: | *стандартный ввод* |
| Имя выходного файла: | *стандартный вывод* |
| Ограничение по времени: | 12 секунд |
| Ограничение по памяти: | 512 мебибайт |

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

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

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

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

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

### Формат входных данных

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

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

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

### Формат выходных данных

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

### Пример

| *стандартный ввод* | *стандартный вывод* |
|---|---|
| 3 4<br>1 1<br>1 2<br>2 1 2<br>1 2<br>3 4<br>3 5<br>3 1 | 1 0 2 1 |

### Пояснение к примеру

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

## Задача I. 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 J. Два пропавших числа

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

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

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

Напишите программу, которая поможет Грише узнать стёртые числа.

### Протокол взаимодействия

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

### Первый запуск

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

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

### Второй запуск

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

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

### Пример

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

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

| *стандартный ввод* | *стандартный вывод* |
|---|---|
| first<br>9 3<br>4 6 1 | 3 4 6 1 |

| *стандартный ввод* | *стандартный вывод* |
|---|---|
| second<br>3 4 6 1<br>9 4<br>5 2 9 8 | 7 3 |