Задача 01. Камилла и язык программирования WR

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Маленькая девочка Камилла любит программировать. Она придумала и реализовала свой собственный язык программирования.

Этот язык называется «WR», и он работает с *wr-строками* — строками, состоящими только из букв «w» и «r». Программа на этом языке — это wr-строка, результат работы программы также является wr-строкой.

В языке WR всего две команды. Команда записи (write) начинается с маленькой буквы «w» и состоит из двух символов: первый из них — сама буква «w», а второй — буква, которую нужно вывести. Команда повтора (repeat) состоит из одного символа — маленькой буквы «r» — и означает, что нужно один раз повторить выполнение предыдущей команды.

При выполнении программа разбивается на команды слева направо, а затем команды выполняются последовательно. Если что-то из этого невозможно, вся программа считается некорректной. Это происходит в двух случаях:

1. Очередная команда начинается с буквы «w», но второго символа в этой команде нет, так как эта буква «w» — последняя буква программы.
2. Команда повтора — первая в программе, поэтому она не может повторить предыдущую команду.

Результат работы некорректной программы — пустая строка.

Камилле нравится, что на её языке можно написать программу, которая выведет другую программу, которую можно тоже выполнить и получить третью программу, и так далее. Девочка хочет последовательно выписать все программы, которые получились при таком процессе. Сначала она выбрала некоторую исходную wr-строку и объявила её *исполняемой*. После этого Камилла выполняет следующий алгоритм:

1. Выписывает исполняемую строку.
2. В результате работы исполняемой строки как программы на языке WR получает новую строку.
3. Если полученная новая строка пуста, заканчивает работу, а иначе объявляет эту новую строку исполняемой и вновь переходит к шагу 1.

По заданной исходной wr-строке выясните, какие строки и в каком порядке выписала Камилла.

Формат входных данных

В единственной строке ввода задана wr-строка — последовательность букв «w» и «r». Она содержит от 1 до 100 букв и не содержит пробелов.

Формат выходных данных

Выведите все строки, которые выписала Камилла, в порядке их выписывания.

Примеры

стандартный ввод	*стандартный вывод*	Пояснение
wwrrwrr	wwrrwrr wwwrr wrr rr	write w, repeat, repeat, write r, repeat write w, write r, repeat write r, repeat repeat <ERROR>
wwrwr	wwrwr wwr ww w	write w, repeat, write r write w, repeat write w write <ERROR>

Пояснения к примерам

В пояснениях справа от примеров показано разбиение каждой wr-строки на команды. Если строка является некорректной программой на языке WR, то команда, вызвавшая проблему, имеет пометку <ERROR> (ошибка).

В первом примере первая строка («wwrrwrr») разбивается на следующие пять команд:

- запись «w»,

- повтор записи «`w`»,
- повтор повтора записи «`w`»,
- запись «`r`»,
- повтор записи «`r`».

Четвёртая строка, полученная в первом примере («`rr`»), начинается с команды повтора, а поэтому является некорректной программой на языке WR.

Во втором примере четвёртая полученная строка («`w`») не разбивается на команды слева направо, а поэтому является некорректной программой на языке WR.

Задача 02. Исправление адресов

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

В почтовой системе Города Роботов адреса записываются как последовательности из девяти десятичных цифр (000 000 000 – 999 999 999). Всего в городе n различных существующих адресов, по которым могут быть доставлены письма.

В главное почтовое отделение Города Роботов прибыли m конвертов. На каждом из конвертов написан адрес, также состоящий из девяти десятичных цифр. Однако некоторые адреса могут быть написаны с ошибками, а некоторые конверты даже могли попасть в почтовое отделение города по ошибке.

Ваша задача — восстановить правильное написание адресов. Считается, что адрес s на конверте *можно исправить* на адрес t в Городе Роботов, если s и t либо совпадают, либо различаются ровно в одной цифре из девяти. Для каждого конверта, адрес на котором исправляется однозначно, выведите исправленный адрес, по которому нужно доставить этот конверт. Если адрес на конверте нельзя исправить ни на какой адрес в городе, поменяв не более чем одну цифру — или, наоборот, существует больше одного адреса в городе, на который его можно исправить — следует также это выяснить.

Формат входных данных

В первой строке ввода задано целое число n — количество различных адресов в Городе Роботов ($1 \leqslant n \leqslant 100$). В следующих n строках записаны сами адреса, по одному на строке. Каждый адрес состоит ровно из девяти десятичных цифр без пробелов.

Следующая строка ввода содержит целое число m — количество конвертов ($1 \leqslant m \leqslant 100$). В следующих m строках записаны адреса, указанные на конвертах, по одному на строке. Каждый адрес также состоит ровно из девяти десятичных цифр без пробелов.

Имейте в виду, что, хотя все n адресов, существующих в Городе Роботов, различны, на нескольких или даже всех m конвертах может быть написан один и тот же адрес.

Формат выходных данных

Выведите m строк. В i-й из этих строк выведите адрес в Городе Роботов, по которому необходимо доставить i-й конверт. Помните, что адрес должен состоять ровно из девяти десятичных цифр без пробелов. Если подходящих адресов нет, выведите вместо адреса число `-1`. Если же подходящих адресов больше одного, выведите вместо адреса число `-2`.

Примеры

стандартный ввод	*стандартный вывод*
2 000000000 999999999 3 012345678 999999989 000000000	-1 999999999 000000000

стандартный ввод	*стандартный вывод*
3 123123123 124123123 134123123 2 124123123 135123123	-2 134123123

Пояснения к примерам

В первом примере (слева):
- в первом адресе на конверте пришлось бы поменять почти все цифры, чтобы получить какой-то из существующих в городе адресов;
- во втором адресе нужно поменять одну цифру;
- в третьем адресе вообще ничего менять не нужно.

Во втором примере (справа):
- первый адрес на конверте может быть исправлен на все три существующих адреса;
- второй адрес восстанавливается однозначно.

Задача G. Вариация Нима

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

На столе лежат n кучек камней: a_1 камней в первой кучке, a_2 камней во второй, ..., a_n в n-й. Двое играют в игру, делая ходы по очереди. За один ход игрок может либо взять произвольное ненулевое количество камней (возможно, все) из одной любой кучки, либо произвольным образом разделить любую существующую кучку, в которой не меньше двух камней, на две непустые кучки. Проигрывает тот, кто не может сделать ход. Кто выигрывает при правильной игре?

Формат входных данных

В первой строке задано целое число t — количество тестов ($1 \leqslant t \leqslant 100$). Следующие t строк содержат сами тесты. Каждая из них начинается с целого числа n — количества кучек ($1 \leqslant n \leqslant 100$). Далее следует n целых чисел a_1, a_2, ..., a_n через пробел — количество камней в кучках ($1 \leqslant a_i \leqslant 10^9$).

Формат выходных данных

Выведите t строк; в i-й строке выведите «FIRST», если в i-м тесте при правильной игре выигрывает первый игрок, и «SECOND», если второй.

Пример

стандартный ввод	*стандартный вывод*
3 1 1 2 1 1 3 1 2 3	FIRST SECOND FIRST

Задача 03. Выражение

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Найдите значение выражения $\sqrt{|a/b + c/d + e/f|}$.

Формат входных данных

В первой строке записаны два целых числа a и b. Во второй строке записаны два целых числа c и d. В третьей строке записаны два целых числа e и f. Здесь $|a|, |c|, |e| \leqslant 10\,000$ и $1 \leqslant b, d, f \leqslant 10\,000$.

Формат выходных данных

Выведите одно вещественное число: значение выражения $\sqrt{|a/b + c/d + e/f|}$. Абсолютная или относительная погрешность должна быть не больше 10^{-9}.

Примеры

стандартный ввод	*стандартный вывод*
1 1 1 1 1 1	1.732050807568877
6 2 -4 3 14 6	2

Задача 04. Одинокий остров

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Остров Одинокий расположен в плоском мире и имеет форму треугольника. Его обитатели могут ходить по всему острову, а также отплывать от него на лодке на расстояние не больше единицы. Рассмотрим фигуру, образованную множеством точек плоскости, которые могут посещать островитяне. Какова площадь этой фигуры?

Вершины треугольника могут лежать на одной прямой или даже совпадать.

Формат входных данных

В первой строке записаны два целых числа x_1 и y_1 — координаты первой вершины треугольника. Во второй строке записаны два целых числа x_2 и y_2 — координаты второй вершины треугольника. В третьей строке записаны два целых числа x_3 и y_3 — координаты третьей вершины треугольника. Гарантируется, что $0 \leqslant x_i, y_i \leqslant 100$.

Формат выходных данных

Выведите одно вещественное число: площадь фигуры. Абсолютная или относительная погрешность должна быть не больше 10^{-9}.

Примеры

стандартный ввод	*стандартный вывод*
0 0 4 5 2 7	28.653253905049
1 1 1 1 1 1	3.14159265358979

Задача 05. Ординальные числа

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Ординальное число — это обобщение понятия натурального числа (здесь и далее в этой задаче 0 считается натуральным числом). Ординальные числа, соответствующие натуральным числам, можно определить рекурсивно:

> Ординальное число, соответствующее натуральному числу n — это множество, элементами которого являются ординальные числа, соответствующие натуральным числам 0, 1, 2, …, $n-1$.

В частности, пустое множество $\varnothing = \{\}$ — ординальное число, соответствующее натуральному числу 0, единице соответствует множество $\{\varnothing\} = \{\{\}\}$, двойке — множество $\{\varnothing, \{\varnothing\}\} = \{\{\}, \{\{\}\}\}$ и так далее. Удобно записывать ординальное число просто как соответствующее ему натуральное число, например, $4 = \{0, 1, 2, 3\}$. Вообще, $n = \{0, 1, \ldots, (n-1)\}$.

Такое определение удобно тем, что его можно расширить на бесконечные множества. К примеру, наименьшее бесконечное ординальное число ω определяется как множество, содержащее все конечные ординальные числа. Следующее за ним число (будем записывать его как $\omega + 1$) равно $\{\omega, 0, 1, \ldots\}$ или, другими словами, $\omega + 1 = \omega \cup \{\omega\}$, и так далее. В общем случае по любому ординальному числу α можно определить следующее ординальное число $\alpha + 1 = \alpha \cup \{\alpha\}$. Однако, предыдущее ординальное число определено не для всех чисел, например, для ω не существует такого γ, что $\omega = \gamma \cup \{\gamma\}$.

Игорь выписал на доске некоторое конечное ординальное число. Запись была корректной и состояла из открывающих и закрывающих фигурных скобок, а также запятых. Формат записи рекурсивно определяется так: запись каждого множества начинается с открывающей фигурной скобки, далее по одному разу следуют записи всех элементов этого множества в произвольном порядке, разделённые запятыми, а в конце стоит закрывающая фигурная скобка. К примеру, ординальное число 0 записывается как «`{}`», 1 — как «`{{}}`», 2 — как «`{{},{{}}}`» или «`{{{}},{}}`» и так далее.

Улучив момент, Владислав подкрался к доске и стёр один символ. Запись больше не является корректной! Это несколько расстраивает Игоря. Установив виновника, Владимир Михайлович попросил Владислава срочно исправить положение. Помогите Владиславу восстановить исходную запись.

Формат входных данных

В первой строке ввода задана запись ординального числа, в которой отсутствует ровно один символ. Длина этой записи — от 1 до $3 \cdot 10^6$ символов.

Формат выходных данных

Выведите одну строку — правильную запись ординального числа, полученную добавлением одного символа в любое место заданной записи. Если правильных ответов несколько, выведите любой из них.

Примеры

стандартный ввод	*стандартный вывод*
`{{{}}{}}`	`{{{}},{}}`
`{{{}},}}`	`{{{}},{}}`
`{{{},{}}`	`{{{}},{}}`

Пояснения к примерам

Во всех трёх тестах после добавления одного символа получается ординальное число 2.

В первом тесте нужно добавить запятую, во втором — открывающую фигурную скобку, в третьем — закрывающую фигурную скобку.

Задача 06. Стресс-тестирование

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Витю пригласили в жюри межпланетной олимпиады по программированию. Это большая честь для него, и он постарался не ударить в грязь лицом. Он реализовал технологию, которая позволяет находить интересные тесты для задач по имеющимся решениям жюри. Эта технология называется *стресс-тестирование*.

Поиск теста осуществляется по следующей схеме. Сначала пишется генератор общего случая теста к задаче, то есть такой генератор, который с достаточно большими вероятностями выдаёт тесты на все случаи, которые необходимо разобрать. Он не обязательно является генератором полностью случайного теста, так как зачастую на таких тестах реализуются далеко не все случаи решения.

Генератор, естественно, использует случайные числа для создания теста. Он инициализируется с помощью функции `randomize`, которая берёт случайное значение для инициализации генератора случайных чисел из внешнего источника (как правило, в качестве такого источника используется системное время). Для того чтобы можно было потом восстановить тест, значение, которым был проинициализирован генератор случайных чисел, выдаётся в файл протокола. Всегда гарантируется, что при одной и той же инициализации генератор случайных чисел будет выдавать одну и ту же последовательность.

После генерации теста на нём запускаются два различных решения жюри. Специальная программа `run` замеряет время работы каждого из решений. Естественно, вся информация о времени работы, параметрах запуска и используемой памяти также выводится в файл протокола работы системы.

Затем происходит сравнение выходных файлов обеих программ. К сожалению, в связи с особенностями используемой суперсовременной операционной системы Doors, информацию о сравнении не так-то просто вывести в файл протокола, поэтому Витя ограничился лишь прерыванием тестирования в случае, если одно из решений выдало неверный ответ.

Весь этот процесс запускается в цикле, пока не будет найден тест, на котором одна из программ выдаёт неверный ответ, или же член жюри, занимающийся тестированием, не сочтёт, что уже было проверено достаточное количество различных тестов.

В первом случае, очевидно, интересным является тест, на котором завалилось решение жюри. Во втором же случае Витя считает интересными тесты, на которых каждое из двух решений работало максимальное время. К сожалению, исходный файл протокола не очень удобен для получения этой информации.

Необходимо написать программу, которая бы преобразовывала файл протокола в более удобный формат, а также находила бы тесты, на которых каждое из решений работало максимальное время.

Формат входных данных

Во входных данных содержится исходный файл протокола. Его размер не превосходит двух мебибайт. Он состоит из одного или нескольких блоков, каждый из которых соответствует запуску обоих решений на одном тесте. Каждый из этих блоков начинается и заканчивается непустой строкой, состоящей из одних символов «`-`».

Внутри блока содержится информация об инициализации генератора случайных чисел, а также о времени работы каждого из решений. Инициализация генератора задаётся строкой в формате «`randseed =` x», где x — целое неотрицательное число, не превосходящее 10^9. За ней следуют протоколы запуска решений. Время работы каждого из решений выведено в формате «`Work time:` x_i `ms`», где x_i — вещественное число, в котором в качестве разделителя целой и дробной части используется запятая — так уж вывела операционная система Doors. Целая часть этого числа не превосходит 10^6. Витя знает, что операционная система Doors всё равно производит измерения таким образом, что дробные части совпадают при совпадении целых частей, поэтому его интересует только целая часть этого числа.

Кроме указанных строк, в файле протокола может содержаться также другая информация, которую Витя пока не хочет анализировать.

Всегда сначала запускается первое решение жюри, а затем второе. Кроме того, запуск решений происходит только после генерации теста, а следовательно, и вывода в файл протокола информации об инициализации генератора случайных чисел.

Формат выходных данных

Для каждого из блоков в порядке их следования должны быть выведены три строки информации. В первой строке выводится значение, которым был проинициализирован генератор случайных чисел, в формате «`At randseed =` x». Во второй и третьей строках выводится время работы первого и второго решений соответственно, в виде «`First:` x_1 `ms`» и «`Second:` x_2 `ms`», где x_i — время работы каждого из решений с отброшенной дробной частью.

Кроме этого, необходимо собрать информацию о том, на каких тестах каждое из решений работало максимальное время. Эта информация выводится после всей информации о блоках. Если какое-то решение работало одинаково долго на нескольких тестах, то выбирается самый первый из блоков, на котором было достигнуто такое время.

Проверка производится автоматически, поэтому вывод вашей программы должен по возможности максимально совпадать с эталонным выводом!

Пример

стандартный ввод

```
------------------------------------------------
randseed = 43794135
Testing ..\b_mb, in=bring.in, out=bring.out, ext=, tl=100000000, ml=100000000 on bring.in
Successful termination (80 ms)
Work time: 80,11520 ms
Maximal memory usage: 10336 Kb
Testing ..\b_al, in=bring.in, out=bring.out, ext=, tl=100000000, ml=100000000 on bring.in
Successful termination (391 ms)
Work time: 390,56160 ms
Maximal memory usage: 2680 Kb
------------------------------------------------
------------------------------------------------
randseed = 43903502
Testing ..\b_mb, in=bring.in, out=bring.out, ext=, tl=100000000, ml=100000000 on bring.in
Successful termination (110 ms)
Work time: 110,15840 ms
Maximal memory usage: 10328 Kb
Testing ..\b_al, in=bring.in, out=bring.out, ext=, tl=100000000, ml=100000000 on bring.in
Successful termination (310 ms)
Work time: 310,44640 ms
Maximal memory usage: 2680 Kb
------------------------------------------------
```

стандартный вывод

```
At randseed = 43794135
First: 80 ms
Second: 390 ms
At randseed = 43903502
First: 110 ms
Second: 310 ms
Maximal work time for first: 110 at randseed = 43903502
Maximal work time for second: 390 at randseed = 43794135
```

Задача 07. Отрезки

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Даны отрезки на прямой. Какое максимальное количество отрезков можно выбрать так, чтобы никакие два из них не пересекались? Отрезки считаются открытыми.

Формат входных данных

В первой строке задано целое число n ($1 \leqslant n \leqslant 100\,000$). В следующих n строках описаны отрезки; i-я из этих строк содержит два целых числа l_i и r_i через пробел — координаты начала и конца отрезка ($1 \leqslant l_i < r_i \leqslant 10^9$).

Формат выходных данных

В первой строке выведите одно число — максимальное количество выбранных отрезков.

Примеры

стандартный ввод	*стандартный вывод*
2 1 2 3 5	2
2 1 3 2 6	1
3 2 4 3 5 1 3	2

Задача 08. Range Variation Query

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

В начальный момент времени последовательность a_n задана следующей формулой: $a_n = n^2 \bmod 12\,345 + n^3 \bmod 23\,456$.

Требуется много раз обрабатывать запросы следующего вида:

- найти разность между максимальным и минимальным значением среди элементов $a_i, a_{i+1}, \ldots, a_j$;
- присвоить элементу a_i значение j.

Формат входных данных

Первая строка содержит целое число k — количество запросов ($1 \leqslant k \leqslant 100\,000$). Следующие k строк содержат запросы, по одному на строке. Запрос номер i описывается двумя целыми числами x_i, y_i.

Если $x_i > 0$, то требуется найти разность между максимальным и минимальным значением среди элементов $a_{x_i} \ldots a_{y_i}$. При этом $1 \leqslant x_i \leqslant y_i \leqslant 100\,000$.

Если $x_i < 0$, то требуется присвоить элементу $a_{|x_i|}$ значение y_i. При этом $-100\,000 \leqslant x_i \leqslant -1$ и $|y_i| \leqslant 100\,000$.

Формат выходных данных

Для каждого запроса первого типа выведите одну строку, содержащую разность между максимальным и минимальным значением на соответствующем отрезке.

Пример

стандартный ввод	*стандартный вывод*
7 1 3 2 4 -2 -100 1 5 8 9 -3 -101 2 3	34 68 250 234 1

Задача 09. Мирные множества

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Группа математиков проводит бои между натуральными числами. Результаты боя между двумя натуральными числами, вообще говоря, случайны, однако подчиняются следующему правилу: если одно из чисел не менее чем в два раза превосходит другое, то большее число всегда побеждает; в противном случае победить может как одно, так и другое число.

Бой называется *неинтересным*, если его результат предопределён. Множество натуральных чисел называется *мирным*, если бой любой пары различных чисел из этого множества неинтересен. *Силой* множества называется сумма чисел в нём. Сколько существует мирных множеств натуральных чисел силы n?

Формат входных данных

В первой строке задано число n ($1 \leqslant n \leqslant 2000$).

Формат выходных данных

В первой строке выведите одно число — количество мирных множеств натуральных чисел силы n.

Примеры

стандартный ввод	*стандартный вывод*
2	1
5	2

Задача 10. Range Minimum Query

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Компания Giggle открывает свой новый офис в Судиславле, и вы пришли на собеседование. Там вам предложили решить следующую задачу.

Создайте структуру данных, которая представляет из себя массив целых чисел. Изначально массив пуст. Структура должна достаточно быстро выполнять операции двух следующих типов:

- запрос: «`?` i j» — вывести значение минимального элемента между i-м и j-м включительно;
- изменение: «`+` i x» — добавить элемент x после i-го элемента массива. Если $i = 0$, то элемент добавляется в начало массива.

Формат входных данных

Первая строка содержит целое число n — число операций над массивом ($1 \leqslant n \leqslant 200\,000$). Следующие n строк описывают сами операции. Все операции добавления являются корректными. Все числа, хранящиеся в массиве, по модулю не превосходят 10^9.

Формат выходных данных

Для каждой операции в отдельной строке выведите её результат.

Пример

стандартный ввод	*стандартный вывод*
8 + 0 5 + 1 3 + 1 4 ? 1 2 + 0 2 ? 2 4 + 4 1 ? 3 5	4 3 1

Задача 11. Число из спичек

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вы умеете выкладывать на столе некоторые цифры от 0 до k с помощью какого-то количества спичек. Ваша задача — построить максимально большое число из n спичек. Например, если вам нужно 6 спичек для цифры 0, 7 спичек для цифры 1 и 8 спичек для цифры 2, то максимальное число, которое можно сложить из 21 спички — это 210.

Формат входных данных

В первой строке даны числа n и k ($1 \leqslant n \leqslant 50$, $0 \leqslant k \leqslant 9$). Во второй строке задано $k+1$ число от 1 до 50 — количество спичек, требуемое на цифры от 0 до k. Гарантируется, что из n спичек можно выложить хотя бы одну из цифр.

Формат выходных данных

Выведите максимальное число, которое можно построить. Лидирующие нули в выводе недопустимы. Можно использовать не все n спичек, а только часть.

Примеры

стандартный ввод
21 2 6 7 8
стандартный вывод
210

стандартный ввод
30 2 5 23 24
стандартный вывод
20

стандартный ввод
1 3 1 5 3 2
стандартный вывод
0

стандартный ввод
50 9 1 1 1 1 1 1 1 1 1 1
стандартный вывод
99999999999999999999999999999999999999999999999999

Задача 12. Покраска в два цвета

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Дан неориентированный граф из n вершин и m рёбер. Нужно покрасить вершины графа в два цвета таким образом, чтобы все рёбра соединяли вершины разных цветов, или выяснить, что это невозможно.

Формат входных данных

В первой строке заданы через пробел два целых числа: число вершин n и число ребёр m ($1 \leqslant n \leqslant 100$, $0 \leqslant m \leqslant 5000$). Далее следует m строк, каждая из которых содержит по два целых числа в диапазоне от 1 до n — номера вершин, которые соединяет ребро.

Формат выходных данных

Если требуемой покраски не существует, выведите одно число `-1`. В противном случае выведите через пробел n целых чисел от 1 до 2 — цвета вершин. Если возможных ответов несколько, выведите любой.

Примеры

стандартный ввод	*стандартный вывод*
3 3 1 2 2 3 3 1	-1
3 1 1 2	1 2 2

Задача 13. Опасный маршрут

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Профессор Дейкстра живёт в очень опасном районе города. Ежедневно бандиты грабят на улицах прохожих. Читая криминальную хронику, профессор вычислил вероятность быть ограбленным при проходе по каждой улице города.

Теперь он хочет найти наиболее безопасный путь от дома до университета, в котором он преподаёт. Иными словами, он хочет найти путь от дома до университета, для которого вероятность быть ограбленным минимальна.

Формат входных данных

В первой строке записаны два числа N и M — количество зданий и количество улиц, соединяющих здания ($1 \leqslant N \leqslant 100$, $1 \leqslant M \leqslant N \cdot (N-1)/2$). В следующей строке находятся числе S и F — номер дома, в котором живёт профессор, и номер дома, в котором находится университет, соответственно. Далее в M строках расположены описания дорог: по целых числа S_i, F_i и P_i — номера зданий, в которых начинается и заканчивается дорога, и вероятность в процентах быть ограбленным, пройдя по дороге, соответственно ($1 \leqslant S_i \leqslant N$, $1 \leqslant F_i \leqslant N$, $0 \leqslant P_i \leqslant 100$, дороги двунаправленные). Гарантируется, что существует хотя бы один путь от дома профессора до университета.

Формат выходных данных

Необходимо вывести одно число — минимальную возможную вероятность быть ограбленным с точностью не менее шести знаков после десятичной точки.

Пример

стандартный ввод	*стандартный вывод*
3 3 1 3 1 2 20 1 3 50 2 3 20	0.36

Пояснение. Для приведённого примера минимальная вероятность быть ограбленным достигается на маршруте $1 - 2 - 3$.

Указание. Пусть вероятность быть ограбленным на пути $A - B$ равна p, а на пути от $B - C$ равна q. Какова вероятность быть ограбленным на пути $A - B - C$?