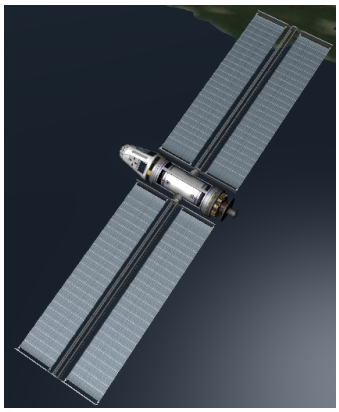
Задача 1. Космический парусник

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Ионные двигатели — перспективный тип двигателей для космических аппаратов. Они имеют очень хороший удельный импульс (отношение тяги к расходу рабочего тела), поэтому, теоретически, позволяют разгонять космические аппараты до очень больших скоростей. Одна из проблем, связанных с такими двигателями, состоит в том, что ионный двигатель, кроме рабочего тела, нуждается еще в электрическом токе. Тяга ионного двигателя пропорциональна электрическому току, который он потребляет. При постоянном напряжении получается, что тяга двигателя пропорциональна мощности источника питания.



Рассмотрим космический корабль с ионным двигателем, питающимся от солнечных батарей. Батареи представляют собой двустороннюю плоскую пластину, освещаемую Солнцем. Их электрическая мощность пропорциональна освещенности, и составляет $P|cos\alpha|$, где P — некоторая константа, пропорциональная площади и КПД батарей, и яркости Солнца, а α — угол между нормалью к плоскости батареи и направлением на Солнце. Двигатель установлен так, что его тяга направлена вдоль плоскости батарей. У такого корабля, максимальная мощность батарей достигается, когда их плоскость перпендикулярна солнечным лучам. К сожалению, при таком положении батарей, корабль может разгоняться только в этой же плоскости. В реальном космическом полете может потребоваться разгон в разных направлениях, и при неудачном выборе направления освещенность батарей может уменьшиться. Если кораблю необходимо разгоняться в направлении от Солнца или на него, освещенность батарей может упасть до нуля. В этом случае, кораблю необходимо выполнять разгон с одним или несколькими поворотами, так что траектория получается похожа на движение парусника галсами.

Вам необходимо написать программу для навигационного компьютера корабля, которая выберет оптимальную по времени траекторию разгона в заданном направлении. Корабль находится в межпланетном пространстве вдали от Солнца и планет, так что направление на Солнце можно считать не меняющимся. Расход рабочего тела за время маневра пренебрежимо мал по сравнению с общей массой корабля, поэтому можно считать, что ускорение линейно пропорционально тяге двигателя, а тяга двигателя, в свою очередь, линейно пропорциональна мощности батарей. Таким образом, ускорение корабля при работающем на полной доступной мощности двигателе определяется по формуле $a_0|cos\alpha|$, где a_0 — некоторая константа, зависящая от константы P, массы корабля, КПД двигателя и других параметров. Иными словами, a_0 – максимально достижимое ускорение при разгоне в плоскости, перпендикулярной направлению на Солнце.

Цель корабля находится на очень большом расстоянии от него, поэтому перемещения корабля во время разгона не играют роли. Время разворота корабля пренебрежимо мало по сравнению с временем разгона. Корабль разгоняется по расписанию, состоящему из отдельных участков. На каждом участке вектор ускорения постоянный. В конце участка корабль разворачивается и продолжает ускорение в другом направлении. Приращение скорости корабля после такого разгона определяется по формуле $\delta\vec{V} = \sum_{i=1}^{n} \vec{a}_i t_i$, где $\vec{a}_i$ — ускорение корабля на i-том участке траектории, а t_i — время работы двигателя на этом участке.

Корабль всегда может развернуться так, чтобы оси батарей были перпендикулярны солнечным лучам. Можно показать, что это позволяет маневрировать только в одной плоскости, проходящей через направление на Солнце и вектор $\delta\vec{V}$. Задача, таким образом, сводится к двумерной.

Требуется найти расписание, обеспечивающее разгон до заданной скорости $\delta\vec{V}$ за минимальное время $\sum_{i=1}^{n} t_i$.

Формат входного файла

В первой строке входного файла записано целое число a_0 — максимально достижимое ускорение корабля при разгоне в плоскости, перпендикулярной Солнцу ($1 \le a_0 \le 100$). Ускорение измеряется в миллиметрах в секунду за секунду.

В следующих строках входного файла заданы параметры требуемых маневров. Каждый маневр описывается двумя целыми числами, записанными через пробел, δV и β, где δV — модуль требуемого приращения скорости в результате маневра, измеряемый в метрах в секунду, а β — угол между требуемым вектором приращения скорости и направлением на Солнце, измеряемый в градусах ($1 \le \delta V \le 10\,000$, $0 \le \beta \le 180$).

Один входной файл содержит описание не более 1000 маневров.

Формат выходного файла

Для каждого маневра в выходной файл нужно вывести строку, содержащую расписание оптимального по времени разгона, обеспечивающего требуемое приращение скорости.

Расписание должно начинаться с целого числа k — количества участков ($1 \le k \le 10$), за которым должны следовать k пар вещественных чисел α_i и t_i, где α_i — угол в градусах между направлением на Солнце и перпендикуляром к плоскости батарей корабля и t_i — время разгона в секундах.

Все числа α_i, t_i рекомендуется выводить с максимально возможной точностью, но не более чем 17 десятичных знаков после точки. Требуемый модуль приращения скорости δV должен достигаться с относительной точностью 10^{-9}, а абсолютная погрешность угла не должна превышать 10^{-7} градуса. При сравнении общего времени разгона с истинно оптимальным относительная погрешность не должна превышать 10^{-6}.

Пример

input.txt	output.txt
100 37 90 10 89	1 90.00 370.00 2 89 100 -271 0.015232804390760

Комментарий

В обоих маневрах оптимально разгоняться прямо на цель.

В первом маневре солнечные батареи повёрнуты наилучшим образом, поэтому время находится простым делением: $\delta V/a_0$.

Во втором маневре солнечные батареи отвёрнуты на один градус от оптимума. В результате время разгона чуть больше, чем $\delta V/a_0$. Появляющаяся из-за неоптимальной работы батарей добавка вынесена во второй участок разгона. Разделять таким образом разгон на участки в вашем решении не требуется.

Ограничений на модуль и знак углов в расписании нет. Угол второго участка равен $-271°$ (что эквивалентно $89°$) в качестве примера.

Задача 2. Копировальный аппарат

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Вам необходимо реализовать функцию автоопределения типа бумаги для копировального аппарата. При использовании этой функции аппарат сканирует документ для определения подходящего типа бумаги из доступных. Необходимо по отсканированному изображению и списку доступных типов бумаги определить, какой из них вместит изображение.

Стороны бумаги параллельны сторонам области сканирования. Левый верхний угол бумаги совмещен с левым верхним углом области сканирования. Бумагу нельзя поворачивать.

Формат входного файла

В первой строке входного файла записаны через пробел целые числа U и V — размеры области сканирования по вертикали и горизонтали ($1 \le U, V \le 100$). В следующих U строках описывается область сканирования. В каждой строке содержится ровно V символов. Часть изображения обозначается символом ‘#’. Отсутствие изображения обозначается символом ‘.’. Гарантируется, что существует хотя бы один символ ‘#’. Расположение области сканирования совпадает с расположением в исходном файле. Левый верхний угол соответствует первому символу первой строки.

В следующей строке записано целое число N — количество доступных типов бумаги ($1 \le N \le 10$). В следующих N строках описываются типы бумаги. В каждой строке записано через пробел по два целых числа y и x — размеры по вертикали и горизонтали ($1 \le y \le U$, $1 \le x \le V$).

Формат выходного файла

В выходной файл необходимо вывести одно целое число — номер подходящего типа бумаги. Типы нумеруются в том порядке, в каком они заданы во входном файле, начиная с единицы. Если подходящих типов несколько, требуется вывести тип с минимальным номером. Гарантируется, что существует хотя бы один подходящий тип бумаги.

Примеры

input.txt	output.txt
4 5 ....#.. .#... ..... 3 2 3 4 3 3 4	2
4 4 .... .#.. #... .... 3 2 2 2 3 3 2	3

Задача 3. Цепная молния

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В игре *Brothers of Sword and Sorcery* есть заклинание «цепная молния». В данной задаче предлагается промоделировать работу этого заклинания.

На поле имеется N юнитов. Также задан неориентированный взвешенный граф, вершины которого соответствуют юнитам, а рёбра определяют способность заклинания перебрасываться с юнита на юнит. Длина ребра определяет, насколько сложно молнии переходить между юнитами, которые соответствуют концам ребра.

У заклинания имееются параметры: сила p, коэффициент затухания k и длительность t. Когда заклинание с этими параметрами применяется к юниту v, то происходит следующая последовательность действий:

1. Если $t = 0$, то заклинание заканчивается.
2. Юниту v наносится урон p.
3. Определяется юнит u, который является ближайшим соседом для v в графе.
4. К юниту u рекурсивно применяется заклинание цепной молнии с силой kp, коэффициентом затухания k и длительностью $t - 1$.

Задан граф юнитов и последовательность применений заклинания. Требуется определить суммарный урон, нанесённый каждому юниту.

Считается, что все юниты на поле достаточно живучи, чтобы устоять после применения всех заклинаний. Гарантируется, что у каждой вершины в графе есть хотя бы один сосед. Также гарантируется, что длины всех рёбер в графе различны. Таким образом, следующий юнит в цепи всегда определяется корректно.

Формат входного файла

В первой строке входного файла записаны через пробел целые числа N, M и K, где N — количество юнитов (вершин в графе), M — количество рёбер в графе и K — количество применений заклинания ($2 \le N \le 10^5$, $\frac{N}{2} \le M \le 10^5$, $1 \le K \le 10^5$).

Во второй строке записано вещественное число k — коэффициент затухания заклинания ($0.0 \le k \le 1.0$), заданное не более чем с шестью знаками после десятичной точки. Коэффициент затухания один и тот же для всех применений заклинания.

В следующих M строках заданы рёбра графа. Каждая строка содержит три целых числа a, b и w ($1 \le a, b \le N$, $a \ne b$, $0 \le w \le 10^9$). Числа a и b являются номерами вершин – концов ребра, а w — длиной ребра. В графе нет кратных рёбер.

В каждой из следующих K строк описано одно применение заклинания. Применение описывается тремя числами v, p и t, где v — номер юнита, к которому применено заклинание, p — сила заклинания и t — длительность заклинания ($1 \le v \le N$, $0.0 \le p \le 1.0$, $1 \le t \le 10^5$). Числа t и v целые. Числа p вещественные, заданы не более чем с шестью знаками после десятичной точки.

Формат выходного файла

В выходной файл необходимо вывести ровно N вещественных чисел, по одному в строке. В i-ой строке должен быть записан суммарный урон, нанесённый i-ому юниту.

Ответ считается верным, если у каждого числа абсолютная или относительная погрешность не превосходит 10^{-7}.

Пример

input.txt	output.txt
4 5 2 0.5 1 2 7 2 3 5 1 3 4 1 4 1 4 3 2 2 0.8 4 1 0.6 3	0.8500000000000 0.8 0.4 0.5

Комментарий

Первое применение наносит урон: 0.8 второму, 0.4 третьему, 0.2 четвёртому, 0.1 первому.
Второе применение наносит урон: 0.6 первому, 0.3 четвёртому, ещё 0.15 первому.

Задача 4. Шоколадки

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

После каждого тура межгалактического турнира по программированию гроссмейстер (и не какой-то там международный, а межгалактический!) Ося Бендер съедал столько шоколадок, какое место он занял.

Потом, как обычно, подробности турнира забылись, а вот шоколадные обертки остались. Все, что он теперь помнит – это сколько было туров и сколько было участников, а их число было одинаково во всех турах. Также он может вычислить сумму занятых им мест. Об этом ему напоминают пустые упаковки из-под шоколада.

Теперь его мучает один вопрос – какое самое высокое место он мог занять на этих турах, и соответственно, какое самое низкое. Естественно, далеко не всегда по имеющимся данным можно однозначно ответить на этот вопрос.

Помогите ему – опрелите самое высокое возможное среди наилучших мест, самое низкое возможное среди наилучших мест, а также самое высокое возможное среди наихудших мест и самое низкое возможное среди наихудших мест.

Формат входного файла

В первой строке входного файла записаны через пробел целые числа N, K и S, где N – количество туров, K – количество участников в каждом туре и S – количество оставленных оберток от шоколадок ($1 \le N, K \le 100, N \le S \le N \cdot K$).

Формат выходного файла

В выходной файл необходимо вывести через пробел четыре целых числа. Первое число – это самое высокое возможное место среди наилучших мест, второе – самое низкое возможное место среди наилучших мест, третье – самое высокое возможное место среди наихудших мест и четвертое – самое низкое возможное место среди наихудших мест.

Пример

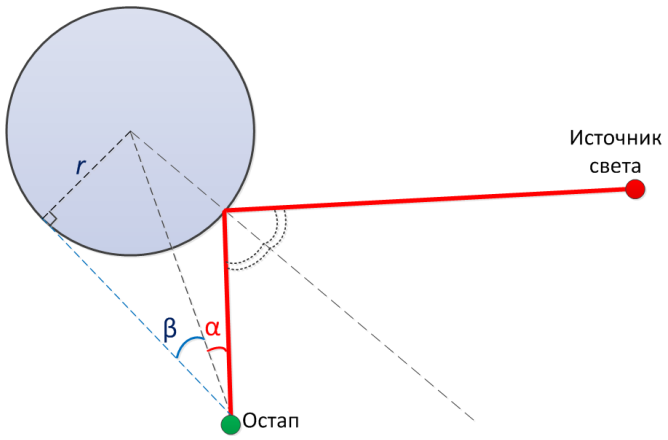
input.txt	output.txt
2 5 7	2 3 4 5

Задача 5. Отражение на шарике

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Астролябия, как известно, измеряет все – было бы что измерять. Ну а мы с ее помощью давайте углы померяем. Тем более, что совсем недавно любимцу Рабиндраната Тагора подарили дивный гаджет – блестящий металлический шар. Теперь этот шар висит в углу комнаты великого комбинатора и все отражает – было бы что отражать. Например, магическую звезду Сириус, или фонарь на улице, или свечку на столе. При этом отражение источника света далеко не всегда находится в центре шара.

Необходимо определить, как далеко от центра шара мы видим отражение источника света. Точнее, необходимо найти отношение угла α к углу β, где α – это угол между направлениями от наблюдателя на центр шара и на отражение его в шаре, а β – угол, под которым мы видим радиус шара.



Формат входного файла

В первой строке входного файла записаны через пробел четыре целых числа x_0, y_0, z_0 и r – координаты центра шара и его радиус. Во второй строке дано три целых числа x_1, y_1 и z_1 – координаты Остапа Бендера, а в третьей строке – три целых числа x_2, y_2 и z_2 , которые задают координаты точечного источника света. Все числа по модулю не превосходят 10^3. Радиус r шара не может быть меньше 10.

Гарантируется, что и Остап и источник света находятся за пределами шара на расстоянии, большем его радиуса. Более того, расстояние от любой точки отрезка, соединяющего Остапа и источнк света, до центра шара больше двойного радиуса шара.

Формат выходного файла

В выходной файл необходимо вывести одно вещественное число, равное α / β с абсолютной или относительной погрешностью не более 10^{-6}.

Пример

input.txt	output.txt
0 0 0 10 1000 0 0 0 1000 0	0.712118480807

Задача 6. Экранная клавиатура

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

С распространением мобильных телефонов с сенсорными экранами стали популярны клавиатуры, использующие метод ввода текста, при котором палец не отрывается от экрана во время набора. Во время ввода слова требуется сделать жест по экранной клавиатуре, который проходит через требуемые клавиши в нужном порядке. Например, если мы хотим набрать слово “python”, то требуемый жест будет выглядеть примерно так, как показано на изображении ниже.

В данной задаче клавиши клавиатуры представляют собой точки на двумерной плоскости, а жест – ломаную на этой плоскости, состоящую из M звеньев. Звенья пронумерованы числами от 1 до M в порядке их появления в жесте.

Пусть дано слово S длины L, выберем для каждой буквы c_i данного слова соответствующую ей точку m_i на некотором звене ломаной с номером k и скажем, что m_i прикреплена к звену k ($1 \le i \le L, 1 \le k \le M$). Назовём набор точек $\{m_1, \ldots, m_L\}$ *разбиением*. Каждая точка m_i должна быть прикреплена ровно к одному звену ломаной, но в то же время к одному звену может быть прикреплено несколько точек. Если в слове несколько раз встречается одна и та же буква, то каждому вхождению буквы будет соответствовать своя точка на ломаной.

Точка, соответствующая каждой следующей букве слова, должна находиться ближе к концу ломаной, чем точка, соответствующая букве, расположенной перед ней, т. е. m_j должна быть ближе к концу ломаной, чем m_i для всех $i < j$. Точка m_j лежит ближе к концу ломаной, чем точка m_i, если выполняется одно из следующих условий:

- точка m_j прикреплена к звену с бóльшим номером, чем m_i;
- точки m_i и m_j прикреплены к одному звену, и расстояние от начала этого звена до m_i не больше, чем до m_j.

Пусть $c_{i,x}$ и $c_{i,y}$ – координаты клавиши буквы c_i, а $m_{i,x}$ и $m_{i,y}$ – кординаты точки m_i, тогда *расстоянием от буквы* c_i до точки m_i будет:

$$dist_i = \sqrt{(c_{i,x} - m_{i,x})^2 + (c_{i,y} - m_{i,y})^2}$$

а *расстоянием слова* S от данного разбиения жеста будет сумма расстояний по всем буквам:

$$dist = \sum_{i=1}^{L} dist_i$$

Назовём *удалённостью слова* S от жеста наименьшее расстояние среди всех возможных разбиений.

Вам дан словарь, состоящий из W различных слов, и жест в виде ломаной. Необходимо найти слово, наиболее соответствующее данной ломаной, т. е., слово с минимальной удалённостью от заданного жеста.

Формат входного файла

В первую строку входного файла записано целое число M – количество звеньев ломаной ($1 \le M \le 20$). Вторая строка содержит два целых числа x_0 и y_0 – координаты начала ломаной. Следующие M строк содержат по два целых числа x_i и y_i – координаты конца i-го звена ломаной. Далее идет строка, в которую записано целое число W – количество слов в словаре ($1 \le W \le 5$). За ней расположены W строк, в каждую из которых записано по одному слову из словаря. Каждое слово состоит только из строчных букв латинского алфавита, и его длина не превосходит 10 символов.

В следующей строке содержится целое число K – количество клавиш на клавиатуре ($1 \le K \le 26$). Далее K строк описывают клавиши клавиатуры. Описание клавиши содержит символ c_j и два целых числа x_j и y_j – букву и координаты j-й клавишы ($'a' \le c_j \le 'z'$). Гарантируется, что все буквы на клавишах попарно различны, и что все буквы, используемые в словах, присутствуют на клавиатуре.

Некоторые клавиши могут иметь одинаковые координаты. Все координаты не превосходят 1000 по абсолютному значению.

Формат выходного файла

В первую строку выходного файла необходимо вывести слово с минимальной удаленностью, во вторую строку – вещественное число, равное расстоянию до этого слова. Если существует несколько слов с минимальной удаленностью, то можно вывести любое из них. В качестве ответа принимаются все слова, абсолютная или относительная прогрешность удалённости которых отличается от наименьшей не более чем на 10^{-6}.

Примеры

input.txt	output.txt
1 1 0 2 0 2 ab ba 2 a 0 2 b 3 2	ab 4.47213595
1 1 0 2 0 1 ba 2 a 0 2 b 3 2	ba 5.00000000

Задача 7. Проблема в аду

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	7 секунд
Ограничение по памяти:	256 мегабайт

Дядя Стёпа играет в компьютерную игру *doomrl*. На этот раз он как никогда близок к тому, чтобы впервые победить силы зла. Однако, простая победа показалась ему недостаточной, и он решил бросить вызов истинной причине адского нашествия — самому Джону Кармаку. Какое это было опрометчивое решение! Так Дядя Стёпа угодил в смертельно опасную ситуацию, из которой может выбраться только с вашей помощью.

Дядя Стёпа находится на уровне *The Lava Pits* один на один с лавовым элементалем, настроенным очень враждебно. У игрока остался всего 1 HP, а патронов и аптечек нет совсем. Единственный шанс спастись — убежать с уровня, не получив никаких повреждений.

Уровень в игре представляется в виде прямоугольного клеточного поля. Клетки считаются соседними, если они граничат либо по стороне, либо по вершине. Клетки уровня могут быть двух типов: с каменным полом или с разлитой лавой. Игрок может ходить только по каменному полу, а лавовый элементать — по любой клетке. Ни игрок, ни элементать не могут выйти за пределы уровня: там находятся непреодолимые каменные стены. В одной из клеток с каменным полом расположен выход с уровня. По этой клетке также могут ходить и игрок, и лавовый элементаль.

Игра пошаговая, игрок и элементаль совершают ходы в некоторой очерёдности. Если скорости игрока и элементаля совпадают, то они ходят по очереди: сначала игрок, потом элементаль, потом снова игрок, и так далее. В данной задаче скорость игрока может быть выше, чем скорость элементаля. Для простоты мы будем рассматривать только тот случай, когда соотношение скоростей $k \geq 1$ — целое. В таком случае сначала ходит игрок k раз, потом элементаль ходит один раз, затем игрок совершает ещё k ходов, затем снова элементаль выполняет свой ход, и так далее.

Игрок в свой ход может:

– остаться в той клетке, в которой он находится;
– переместиться в одну из соседних клеток.

Если игрок в какой-то момент попадает в клетку, в которой расположен выход с уровня, то он сразу же убегает живым и почти счастливым. Переместиться в клетку с лавовым элементалем игрок не может, даже если в этой клетке находится выход с уровня.

Лавовый элементать в свой ход может:

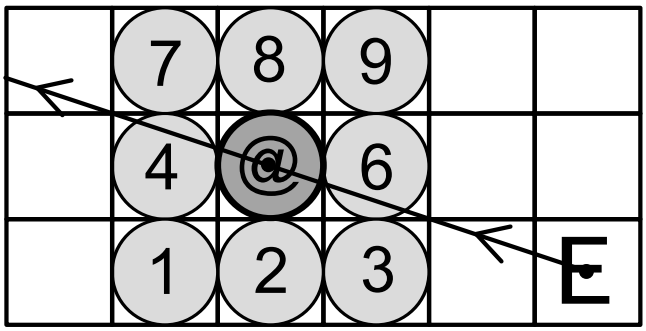
– остаться на месте;
– переместиться в одну из соседних клеток;
– ударить игрока, если он находится в соседней клетке (от этого игрок умирает);
– запустить в игрока огненный шар.

Последняя способность особенно опасна, ведь одно попадание — и игра проиграна. Огненный шар вылетает из центра клетки, в которой находится лавовый элементаль, в центр клетки-цели. Шар летит по прямой. С точки зрения игровой механики шар пролетает весь свой путь мгновенно. Геометрически игрок считается кругом, вписанным в клетку, в которой он стоит. Если луч, являющийся траекторией огненного шара, пересекает этот круг, то с

некоторой вероятностью игрок погибает. Дядя Стёпа не хочет полагаться на случай, поэтому для него недопустима ситуация, в которой он может с какой-то вероятностью умереть.

К счастью, у Дяди Стёпы есть умение *Dodgemaster*. Благодаря этому умению лавовый элементаль всегда запускает огненный шар в ту клетку, на которой находился игрок перед тем, как совершил свой последний ход. Обратите внимание, что эта клетка-цель либо является соседней для той клетки, на которой стоит игрок, либо совпадает с ней. Таким образом, если игрок будет перемещаться примерно перпендикулярно направлению на лавового элементаля, то он сможет гарантированно уклоняться от его огненных шаров.

На картинке ниже показан пример действия умения. Допустим, сейчас ходит игрок (`@`), а сразу после этого будет ходить элементаль (`E`). Тогда элементаль на своём ходу сможет запустить огненный шар в ту клетку, на которой сейчас стоит игрок. Игрок вынужден перейти в одну из соседних клеток, обозначенных цифрами 7, 8, 9, 1, 2, 3. Если он останется на месте (5) или перейдёт в клетку 4 или 6, то элементаль сможет в него попасть.



Дядя Стёпа знает, в каких клетках изначально расположены игрок, лавовый элементаль, выход с уровня. Нужно помочь ему составить план бегства. Он настолько напуган, что в процессе бегства не сможет обращать внимание ни на действия огненного элементаля, ни на пролетающие мимо огненные шары, ни на что-либо ещё. Поэтому требуется найти такую последовательность действий игрока, которая гарантированно приведёт его к спасению, независимо от того, как будет ходить элементаль.

На текущий момент соотношение скоростей игрока и элементаля равно k_{min}. Однако дядя Стёпа согласен продать душу дьяволу — он готов считерить! Вам разрешается предварительно ускорить игрока, повысив соотношение скоростей. Читы не всесильны: повысить скорость выше k_{max} нельзя. Дядя Стёпа просит найти минимальное соотношение скоростей из этого диапазона, при котором возможен гарантированный побег.

Формат входного файла

В первой строке входного файла записано четыре целых числа m, n, k_{min} и k_{max}, где m и n — размеры прямоугольного поля, k_{min}, k_{max} — диапазон возможных значений соотношения скоростей ($1 \le m, n \le 250$, $1 \le k_{min} \le k_{max} \le 25$).

В следующих m строках задана конфигурация уровня. Каждая из них содержит ровно n символов. Каждый символ определяет исходное состояние клетки:

- `.` — пустая клетка с каменным полом;
- `=` — пустая клетка с лавой;
- `E` — клетка с каменным полом, в которой изначально стоит элементаль;
- `@` — клетка с каменным полом, в которой изначально стоит игрок;
- `>` — пустая клетка с каменным полом, в которой находится выход с уровня.

Гарантируется, что на поле ровно одна клетка каждого из типов `E`, `@`, `>`.

Формат выходного файла

Если гарантированно выбраться с уровня нельзя ни при каком допустимом ускорении игрока, то в единственной строке выходного файла должно быть записано слово `DOOMED`.

Иначе необходимо вывести в первую строку минимально возможное соотношение скоростей ускоренного игрока и элементаля $k_{upd} \in [k_{min}, k_{max}]$. Если ускорять игрока не требуется, то должно быть выведено исходное соотношение скоростей k_{min}.

Во вторую строку нужно вывести последовательность действий игрока, приводящую к спасению. Каждое действие обозначается одним символом — цифрой от `1` до `9`, соответствующей направлению движения на `NumPad`. Более подробно:

`5` — остаться на месте;

`8`, `2`, `6`, `4` — переместиться вверх, вниз, вправо, влево соответственно;

`1`, `3`, `9`, `7` — переместиться по диагонали влево-вниз, вправо-вниз, вправо-вверх и влево-вверх соответственно.

Если при минимальном соотношении скоростей существует несколько спасительных последовательностей действий игрока, требуется выбрать среди них самую короткую. Если таких вариантов по-прежнему несколько, нужно вывести любой из них.

Примеры

input.txt	output.txt
3 7 1 4 ======= E@....> =======	DOOMED
3 6 1 25 @.=.=. =..... E===.>	2 36636

Комментарий

В первом примере игроку нужно сделать пять ходов, чтобы добежать до выхода. Однако $k_{max} = 4$, значит, как минимум, один раз сходит элементаль. Он может метнуть в игрока огненный шар, а уклониться некуда, потому что кругом лава.

Во втором примере можно убежать при $k_{upd} = 2$. Варианты кратчайших спасительных последовательностей действий: `36636`, `63636`, `36933`. Последовательность `36663` не является спасительной, потому что элементаль может подстрелить игрока, сначала переместившись по направлению 8, а потом метнув в него шар.

Задача 8. Линейные уравнения

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

На вход вашей программе подаётся система из двух линейных уравнений с двумя неизвестными x и y. Необходимо решить эту систему. Гарантируется, что существует ровно одно решение.

Уравнения задаются своей текстовой записью. Каждое уравнение состоит из двух алгебраических выражений, разделённых знаком `'='` (знак равенства, ASCII 61). Каждое алгебраическое выражение состоит из нескольких слагаемых, разделённых знаками `'+'` и `'-'` (ASCII 43 и 45). В самом начале выражения может стоять унарный минус `'-'` (ASCII 45). Каждое слагаемое состоит либо из одного целого числа, либо из двух множителей: левого и правого. Левым множителем может быть только целое число. Правым множителем может быть либо переменная, либо алгебраическое выражение, заключенное в скобки `'('` и `')'` (ASCII 40 и 41). Переменные представляются латинскими буквами `'x'` и `'y'` (ASCII 120 и 121 соответственно).

Формат входного файла

Входной файл состоит из двух строк. В каждой строке приведено по одному уравнению. Длина каждой строки не превышает 100. Гарантируется, что строки не содержат пробелов, символов табуляции и прочих, не описанных в условии, символов. Также гарантируется, что после раскрытия скобок коэффициенты при неизвестных не превышают по модулю 10^5.

Формат выходного файла

В выходной файл необходимо вывести решение заданной системы уравнений. В первую строку выдать значение переменной x, а во вторую – значение переменной y. Значения выдавать с абсолютной или относительной погрешностью не больше 10^{-3}.

Пример

input.txt	output.txt
2(x+1+2y)+5y=1x 2y=4	-20 2

Задача 9. Солнечный день

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Гоша работает программистом в Неизвестной софтверной конторе. Гоша — сова. Он любит спать днём и работать после полуночи. А может быть, даже вампир... Ведь Гоша проводит большую часть жизни в «пещерах» и не выносит прямого солнечного света. И коллеги постоянно на него жалуются: «Всю кровь высосал, ... !»

Однажды жарким летом Гоша с вечера задержался на работе. А когда закончились печенюшки, был уже полдень. Мобильный он забыл дома, деньги тоже. Гоше очень нужно добраться до дома, и не осталось другого выбора, кроме как идти пешком. Но на улице его ждёт заклятый враг — солнце!

Карта города представляется в виде набора перекрестков, соединённых дорогами. Работа и дом Гоши расположены на разных перекрестках. По дорогам можно ходить в обоих направлениях. Каждая дорога частично освещена солнечным светом, частично находится в тени. В данной задаче будем считать, что зоны освещённости не изменяются с течением времени.

Гоша ходит с постоянной скоростью один метр в секунду. Он вправе ходить по дорогам как душа пожелает: выбирать любое направление на перекрестке, разворачиваться в любой момент. Однако он не может выдержать прямого солнечного света более T секунд подряд. Чтобы пройти большое расстояние, ему необходимо периодически заходить в тень, чтобы остыть. Сколько времени Гоша остывает в тени — не имеет значения. Он может пройти весь теневой участок, а может просто постоять в тени.

Требуется определить оптимальный маршрут между домом Гоши и работой. Гоша очень устаёт от ходьбы, поэтому требуется минимизировать суммарное расстояние, которое ему придётся пройти.

Формат входного файла

В первой строке входного файла записаны через пробел пять целых чисел N, M, T, A и B, где N — количество перекрестков в городе, M — количество дорог в городе, T — сколько секунд Гоша выдерживает на солнце, A — номер перекрестка, на котором расположена контора Гоши, B — номер перекрестка, на котором расположен дом Гоши ($1 \le N \le 100$, $1 \le M \le \frac{N(N-1)}{2}$, $1 \le T \le 150$, $1 \le A \ne B \le N$).

В следующих M строках заданы дороги, по одной на строке. В строку, содержащую описание одной дороги, записаны через пробел целые числа: $u, v, k, l_1, t_1, l_2, t_2, \ldots, l_k, t_k$, где u — номер перекрестка, от которого начинается дорога, v — номер перекрестка, в котором заканчивается дорога, k — количество участков дороги, l_i — длина i-ого участка в метрах, t_i — тип освещённости i-ого участка ($1 \le u \ne v \le N$, $k \ge 1$, $1 \le l_i \le 5\,000$, $t_i \in \{0, 1\}$). Если тип освещённости равен 1, то участок полностью освещён солнцем, а если 0 — то участок полностью затенён. Участки дороги задаются подряд от начала к концу в порядке описания.

Общее количество участков в городе не превосходит 200 000. Между каждыми двумя перекрестками не больше одной дороги. Нет дорог, соединяющих перекресток сам с собой.

Формат выходного файла

В выходной файл нобходимо вывести одно целое число — минимально возможное расстояние, которое требуется пройти Гоше, чтобы добраться до дома. Если добраться согласно условию задачи до дома нельзя, то требуется вывести число -1.

Пример

input.txt	output.txt
6 9 5 2 5 1 2 1 5 1 1 3 3 3 0 3 1 3 0 2 3 2 100 0 2 1 6 2 1 1 0 4 6 1 3 1 4 3 1 7 1 3 5 1 4 1 4 5 2 3 1 1 0 6 5 1 6 1	18

Комментарий

В тесте из условия оптимальный маршрут проходит перекрестки $2, 1, 3, 5$ и имеет длину 18. Другой допустимый маршрут проходит через перекрестки $2, 3, 5$, однако он длиннее (106). Самый короткий маршрут идёт через перекрестки $2, 6, 5$, однако в нём есть освещённый участок длиной 6, который Гоша преодолеть не может.