Задача 5. Миша и негатив

Имя входного файла:	negative.in
Имя выходного файла:	negative.out
Максимальное время работы на одном тесте:	2 секунды
Максимальный объем используемой памяти:	64 мегабайта
Максимальная оценка:	100 баллов

Миша уже научился хорошо фотографировать и недавно увлекся программированием. Первая программа, которую он написал, позволяет формировать негатив бинарного черно-белого изображения.

Бинарное черно-белое изображение – это прямоугольник, состоящий из пикселей, каждый из которых может быть либо черным, либо белым. Негатив такого изображения получается путем замены каждого черного пикселя на белый, а каждого белого пикселя – на черный.

Миша, как начинающий программист, написал свою программу с ошибкой, поэтому в результате ее исполнения мог получаться некорректный негатив. Для того чтобы оценить уровень несоответствия получаемого негатива изображению, Миша начал тестировать свою программу.

В качестве входных данных он использовал исходные изображения. Сформированные программой негативы он начал тщательно анализировать, каждый раз определяя число пикселей негатива, которые получены с ошибкой.

Требуется написать программу, которая в качестве входных данных использует исходное бинарное черно-белое изображение и полученный Мишиной программой негатив, и на основе этого определяет количество пикселей, в которых допущена ошибка.

Формат входных данных

Первая строка входного файла содержит целые числа n и m ($1 \le n, m \le 100$) – высоту и ширину исходного изображения (в пикселях).

Следующие n строк содержат описание исходного изображения. Каждая строка состоит из m символов «B» и «W». Символ «B» соответствует черному пикселю, а символ «W» – белому.

Далее следует пустая строка, а после нее – описание изображения, выведенного Мишиной программой в том же формате, что и исходное изображение.

Формат выходных данных

В выходной файл необходимо вывести число пикселей негатива, которые неправильно сформированы Мишиной программой.

Примеры входных и выходных данных

negative.in	negative.out
3 4 WBBW BBBB WBBW BWWW WWWB BWWB	2
2 2 BW BB WW BW	2

Задача 6. Треугольник Максима

Имя входного файла:	triangle.in
Имя выходного файла:	triangle.out
Максимальное время работы на одном тесте:	2 секунды
Максимальный объем используемой памяти:	64 мегабайта
Максимальная оценка:	100 баллов

С детства Максим был неплохим музыкантом и мастером на все руки. Недавно он самостоятельно сделал несложный перкуссионный музыкальный инструмент – треугольник. Ему нужно узнать, какова частота звука, издаваемого его инструментом.

У Максима есть профессиональный музыкальный тюнер, с помощью которого можно проигрывать ноту с заданной частотой. Максим действует следующим образом: он включает на тюнере ноты с разными частотами и для каждой ноты на слух определяет, ближе или дальше она к издаваемому треугольником звуку, чем предыдущая нота. Поскольку слух у Максима абсолютный, он определяет это всегда абсолютно верно.

Вам Максим показал запись, в которой приведена последовательность частот, выставляемых им на тюнере, и про каждую ноту, начиная со второй, записано – ближе или дальше она к звуку треугольника, чем предыдущая нота. Заранее известно, что частота звучания треугольника Максима составляет не менее 30 герц и не более 4000 герц.

Требуется написать программу, которая определяет, в каком интервале может находиться частота звучания треугольника.

Формат входных данных

Первая строка входного файла содержит целое число n – количество нот, которые воспроизводил Максим с помощью тюнера ($2 \le n \le 1000$). Последующие n строк содержат записи Максима, причем каждая строка содержит две компоненты: вещественное число f_i – частоту, выставленную на тюнере, в герцах ($30 \le f_i \le 4000$), и слово «closer» или слово «further» для каждой частоты кроме первой.

Слово «closer» означает, что частота данной ноты ближе к частоте звучания треугольника, чем частота предыдущей ноты, что формально описывается соотношением: $|f_i - f_{\text{треуг.}}| < |f_{i-1} - f_{\text{треуг.}}|$.

Слово «further» означает, что частота данной ноты дальше, чем предыдущая.

Если оказалось, что очередная нота так же близка к звуку треугольника, как и предыдущая нота, то Максим мог записать любое из двух указанных выше слов.

Гарантируется, что результаты, полученные Максимом, непротиворечивы.

Формат выходных данных

В выходной файл необходимо вывести через пробел два вещественных числа – наименьшее и наибольшее возможное значение частоты звучания треугольника, изготовленного Максимом.

Примеры входных и выходных данных

triangle.in	triangle.out
3 440.0 220.0 closer 300.0 further	30.0 260.0
4 554.0 880.0 further 440.0 closer 622.0 closer	531.0 660.0

Система оценивания

Решения, правильно работающие только для целых чисел f_i, имеющих одинаковую четность, будут оцениваться из 40 баллов.

Задача 7. Производство деталей

Имя входного файла:	details.in
Имя выходного файла:	details.out
Максимальное время работы на одном тесте:	2 секунды
Максимальный объем используемой памяти:	64 мегабайта
Максимальная оценка:	100 баллов

Предприятие «Авто-2010» выпускает двигатели для известных во всем мире автомобилей. Двигатель состоит ровно из n деталей, пронумерованных от 1 до n, при этом деталь с номером i изготавливается за p_i секунд. Специфика предприятия «Авто-2010» заключается в том, что там одновременно может изготавливаться лишь одна деталь двигателя. Для производства некоторых деталей необходимо иметь предварительно изготовленный набор других деталей.

Генеральный директор «Авто-2010» поставил перед предприятием амбициозную задачу – за наименьшее время изготовить деталь с номером 1, чтобы представить ее на выставке.

Требуется написать программу, которая по заданным зависимостям порядка производства между деталями найдет наименьшее время, за которое можно произвести деталь с номером 1.

Формат входных данных

Первая строка входного файла содержит число n ($1 \le n \le 100000$) – количество деталей двигателя. Вторая строка содержит n натуральных чисел p_1, p_2 … p_n , определяющих время изготовления каждой детали в секундах. Время для изготовления каждой детали не превосходит 10^9 секунд.

Каждая из последующих n строк входного файла описывает характеристики производства деталей. Здесь i-ая строка содержит число деталей k_i, которые требуются для производства детали с номером i, а также их номера. Сумма всех чисел k_i не превосходит 200000.

Известно, что не существует циклических зависимостей в производстве деталей.

Формат выходных данных

В первой строке выходного файла два числа: минимальное время (в секундах), необходимое для скорейшего производства детали с номером 1 и число k деталей, которые необходимо для этого произвести. Во второй строке выведите через пробел k чисел – номера деталей в том порядке, в котором следует их производить для скорейшего производства детали номер 1.

Примеры входных и выходных данных

details.in	details.out
3 100 200 300 1 2 0 2 2 1	300 2 2 1
2 2 3 1 2 0	5 2 2 1
4 2 3 4 5 2 3 2 1 3 0 2 1 3	9 3 3 2 1

Система оценивания

Решения, правильно работающие только для тестов, в которых n не превосходит 10, будут оцениваться из 40 баллов.

Решения, правильно работающие только для тестов, в которых n не превосходит 100, будут оцениваться из 60 баллов.

Решения, правильно работающие только для тестов, в которых n не превосходит 1000, будут оцениваться из 80 баллов.

Задача 8. Новое слово в рекламе

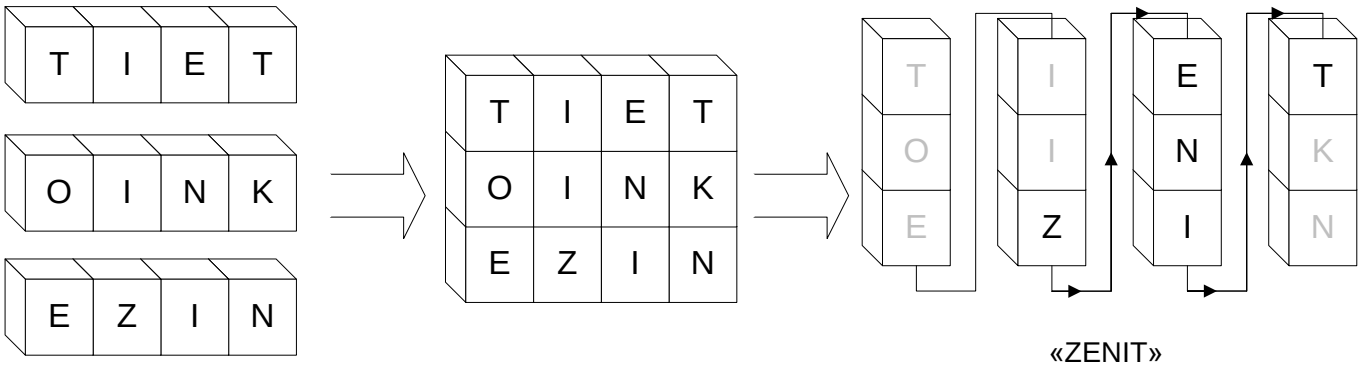
Имя входного файла:	word.in
Имя выходного файла:	word.out
Максимальное время работы на одном тесте:	2 секунды
Максимальный объем используемой памяти:	64 мегабайта
Максимальная оценка:	100 баллов

В наши дни предоставление поверхностей заборов и стен промышленных зданий рекламодателям – уже не оригинальный способ получить дополнительный заработок, а нечто само собой разумеющееся.

Небольшая компания «Домострой» также решила выйти на этот рынок и стала предлагать место для рекламы на своих блоках заборов. Блок представляет собой параллелепипед размером $1\times1\times L$, на одной из сторон которого есть место для рекламы – пространство размера $1\times L$, в которое можно вписать ровно L букв латинского алфавита.

К сожалению, иногда сделки у компании срывались, и заранее подготовленные блоки с рекламой отправлялись на склад. Со временем там скопилось приличное количество блоков различных типов (блоки разных типов отличаются друг от друга только надписью), поэтому было решено использовать их вторично.

Была предложена следующая идея: если поставить несколько блоков друг на друга, то, читая сверху вниз и слева направо, можно будет прочитать какой-нибудь другой текст, как показано на рисунке.



Таким образом, можно получить рекламную надпись для нового клиента. При этом из эстетических соображений при прочтении конечной надписи разрывы в виде закрашенных букв недопустимы.

Опишем этот процесс более формально. После того, как некоторое число K блоков, каждый из которых имеет длину L, поставили друг на друга, получилась прямоугольная таблица размером $K\times L$, в каждой клетке которой находится буква латинского алфавита. Каждый рекламный блок соответствует строке этой таблицы. Теперь содержимое этой таблицы выписывается по столбцам, начиная с самого левого. При этом в каждом столбце буквы выписываются сверху вниз. В случае, изображенном на рисунке, в результате этого процесса получилась бы строка «TOEIIZENITKN». Необходимо, чтобы рекламная надпись, необходимая заказчику, входила в получившуюся строку как подстрока: «TOEII**ZENIT**KN».

Требуется написать программу, которая будет определять, какое минимальное количество блоков надо использовать, чтобы получить рекламную надпись, необходимую заказчику. При этом можно считать, что на складе блоков каждого типа неограниченно много.

Формат входных данных

Первая строка входного файла входного файла содержит два натуральных числа N и L – число различных типов блоков на складе и длина каждого блока соответственно ($1 \le N \le 100$, $1 \le L \le 100$). Последующие N строк содержат по одной записи длиной L, состоящей из строчных

латинских букв – надписи на блоках соответствующего типа. Надписи на блоках разных типов не совпадают.

Последняя строка входного файла содержит новую рекламную надпись s – строку, состоящую только из строчных латинских букв ($1 \le |s| \le 200$). Можно считать, что на складе находится неограниченное число блоков каждого типа.

Формат выходных данных

В первой строке выходного файла необходимо вывести натуральное число K – минимальное количество блоков, которое нужно использовать для составления новой рекламы. Следующая строка должна содержать K чисел – номера типов блоков, которые нужно для этого использовать, перечисляя их сверху вниз. Типы блоков нумеруются с единицы в порядке их задания во входном файле.

Если ответов несколько, выведите любой из них. Если решения не существует, выведите в выходной файл число –1.

Примеры входных и выходных данных

word.in	word.out
3 4 tiet oink ezin zenit	3 1 2 3
2 11 sillysample happysample sam	1 2
2 3 baa aab bb	1 2 2
2 3 aaa bbb cc	-1

Система оценивания

Решения, правильно работающие только для $N \le 5$, будут оцениваться из 50 баллов.