

## Задача 1. Соревнование картингистов

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

После очередного этапа чемпионата мира по кольцевым автогонкам на автомобилях с открытыми колесами Формула-А гонщики собрались вместе в кафе, чтобы обсудить полученные результаты. Они вспомнили, что в молодости соревновались не на больших болидах, а на картах – спортивных автомобилях меньших размеров.

Друзья решили выяснить победителя в одной из гонок на картах. Победителем гонки являлся тот гонщик, у которого суммарное время прохождения всех кругов трассы было минимальным.

Поскольку окончательные результаты не сохранились, то каждый из  $n$  участников той гонки вспомнил и выписал результаты прохождения каждого из  $m$  кругов трассы. К сожалению, по этой информации гонщикам было сложно вычислить победителя той гонки. В связи с этим они попросили сделать это вас.

**Требуется** написать программу, которая вычислит победителя гонки на картах, о которой говорили гонщики.

### Формат входных данных

Первая строка входного файла содержит два целых числа  $n$  и  $m$  ( $1 \leq n, m \leq 100$ ). Последующие  $2 \cdot n$  строк описывают прохождение трассы каждым из участников. Описание прохождения трассы участником состоит из двух строк. Первая строка содержит имя участника с использованием только латинских букв (строчных и заглавных). Имена всех участников различны, строчные и заглавные буквы в именах различаются.

Вторая строка содержит  $m$  положительных целых чисел, где каждое число – это время прохождения данным участником каждого из  $m$  кругов трассы (каждое из этих чисел не превосходит 1000). Длина каждой строки не превышает 255 символов.

### Формат выходных данных

В выходной файл необходимо вывести имя победителя гонки на картах. Если победителей несколько, требуется вывести имя любого из них.

### Пример входных и выходных данных

race.in	race.out
5 3 Sumaher 2 1 1 Barikelo 2 1 2 Olonso 1 2 1 Vasya 1 1 1 Fedya 1 1 1	Fedya

## Задача 2. Дипломы

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

Когда Петя учился в школе, он часто участвовал в олимпиадах по информатике, математике и физике. Так как он был достаточно способным мальчиком и усердно учился, то на многих из этих олимпиад он получал дипломы. К окончанию школы у него накопилось  $n$  дипломов, причем, как оказалось, все они имели одинаковые размеры:  $w$  – в ширину и  $h$  – в высоту.

Сейчас Петя учится в одном из лучших российских университетов и живет в общежитии со своими одногруппниками. Он решил украсить свою комнату, повесив на одну из стен свои дипломы за школьные олимпиады. Так как к бетонной стене прикрепить дипломы достаточно трудно, то он решил купить специальную доску из пробкового дерева, чтобы прикрепить ее к стене, а к ней – дипломы. Для того чтобы эта конструкция выглядела более красиво, Петя хочет, чтобы доска была квадратной и занимала как можно меньше места на стене. Каждый диплом должен быть размещен строго в прямоугольнике размером  $w$  на  $h$ . Прямоугольники, соответствующие различным дипломам, не должны иметь общих внутренних точек.

**Требуется** написать программу, которая вычислит минимальный размер стороны доски, которая потребуется Пете для размещения всех своих дипломов.

### Формат входных данных

Входной файл содержит три целых числа:  $w, h, n$  ( $1 \leq w, h, n \leq 10^9$ ).

### Формат выходных данных

В выходной файл необходимо вывести ответ на поставленную задачу.

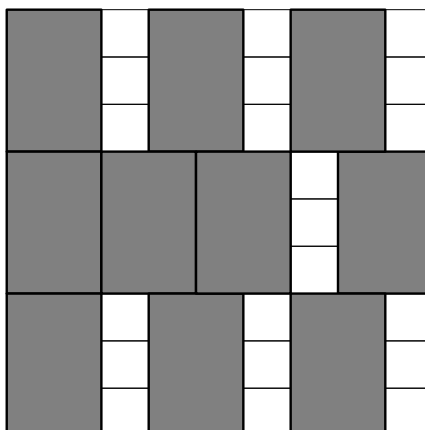
### Пример входных и выходных данных

diploma.in	diploma.out
2 3 10	9

### Система оценивания

Решения, правильно работающие только при  $w, h, n \leq 1000$ , будут оцениваться из 40 баллов.

### Иллюстрация к примеру



### Задача 3. Булева функция

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

Недавно на уроке информатики ученики одного из классов изучили булевы функции. Напомним, что булева функция  $f$  сопоставляет значениям двух булевых *аргументов*, каждый из которых может быть равен 0 или 1, третье булево значение, называемое *результатом*. Для учеников, которые выразили желание более подробно изучать эту тему, учительница информатики на дополнительном уроке ввела в рассмотрение понятие *цепного вычисления* булевой функции  $f$ .

Если задана булева функция  $f$  и набор из  $N$  булевых значений  $a_1, a_2, \dots, a_N$ , то *результат цепного вычисления* этой булевой функции определяется следующим образом:

- если  $N = 1$ , то он равен  $a_1$ ;
- если  $N > 1$ , то он равен результату цепного вычисления булевой функции  $f$  для набора из  $(N-1)$  булевого значения  $f(a_1, a_2), a_3, \dots, a_N$ , который получается путем замены первых двух булевых значений в наборе из  $N$  булевых значений на единственное булево значение – результат вычисления функции  $f$  от  $a_1$  и  $a_2$ .

Например, если изначально задано три булевых значения:  $a_1 = 0, a_2 = 1, a_3 = 0$ , а функция  $f$  – *ИЛИ* (OR), то после первого шага получается два булевых значения –  $(0 \text{ OR } 1)$  и  $0$ , то есть, 1 и 0. После второго (и последнего) шага получается результат цепного вычисления, равный 1, так как  $1 \text{ OR } 0 = 1$ .

В конце дополнительного урока учительница информатики написала на доске булеву функцию  $f$  и попросила одного из учеников выбрать такие  $N$  булевых значений  $a_i$ , чтобы результат цепного вычисления этой функции был равен единице. Более того, она попросила найти такой набор булевых значений, в котором число единиц было бы *как можно большим*.

**Требуется** написать программу, которая решала бы поставленную учительницей задачу.

#### Формат входных данных

Первая строка входного файла содержит одно натуральное число  $N$  ( $2 \leq N \leq 100\,000$ ).

Вторая строка входного файла содержит описание булевой функции в виде четырех чисел, каждое из которых – ноль или единица. Первое из них есть результат вычисления функции в случае, если оба аргумента – нули, второе – результат в случае, если первый аргумент – ноль, второй – единица, третье – результат в случае, если первый аргумент – единица, второй – ноль, а четвертый – в случае, если оба аргумента – единицы.

#### Формат выходных данных

В выходной файл необходимо вывести строку из  $N$  символов, определяющих искомый набор булевых значений  $a_i$  с максимально возможным числом единиц. Если ответов несколько, требуется вывести любой из них. Если такого набора не существует, выведите в выходной файл фразу «No solution».

#### Пример входных и выходных данных

function.in	function.out
4 0110	1011
5 0100	11111
6 0000	No solution

**Пояснения к примерам**

В первом примере процесс вычисления цепного значения булевой функции  $f$  происходит следующим образом:

$$1011 \rightarrow 111 \rightarrow 01 \rightarrow 1$$

Во втором примере вычисление цепного значения булевой функции  $f$  происходит следующим образом:

$$11111 \rightarrow 0111 \rightarrow 111 \rightarrow 01 \rightarrow 1$$

В третьем примере получить цепное значение булевой функции  $f$ , равное 1, невозможно.

**Система оценивания**

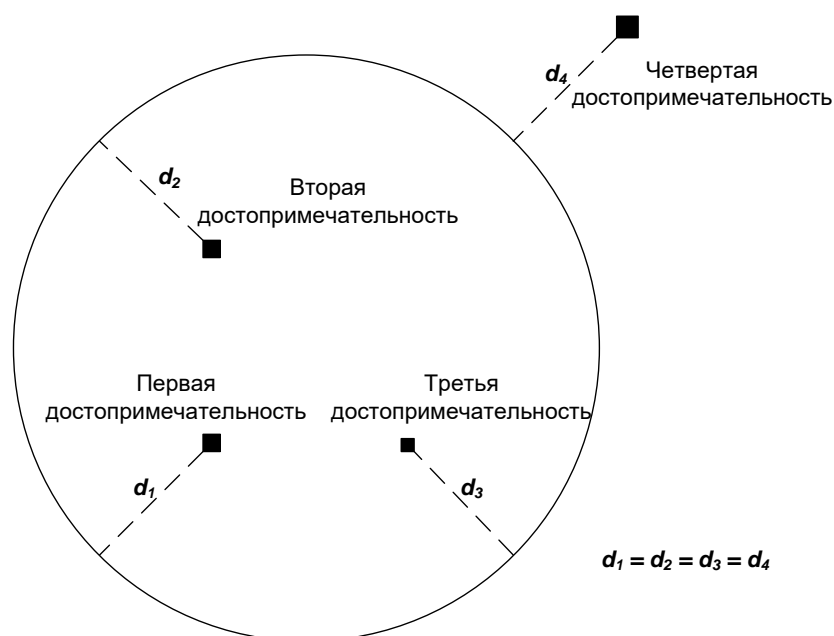
Решения, правильно работающие только при  $N \leq 20$ , будут оцениваться из 40 баллов.

## Задача 4. Кольцевая автодорога

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

К 2110 году город Флэтбург, являясь одним из крупнейших городов мира, не имеет обходной автомагистрали, что является существенным препятствием для его развития как крупнейшего транспортного центра мирового значения. В связи с этим еще в 2065 году при разработке Генерального плана развития Флэтбурга была определена необходимость строительства кольцевой автомобильной дороги.

В Генеральном плане также были обозначены требования к этой дороге. Она должна соответствовать статусу кольцевой – иметь форму окружности. Кроме этого, четыре крупные достопримечательности Флэтбурга должны быть в одинаковой транспортной доступности от дороги. Это предполагается обеспечить тем, что они будут находиться на равном расстоянии от нее. Расстоянием от точки расположения достопримечательности до дороги называется *наименьшее* из расстояний от этой точки до некоторой точки, принадлежащей окружности автодороги.



Дирекция по строительству города Флэтбурга, ответственная за постройку кольцевой автодороги, решила привлечь передовых программистов для выбора оптимального плана постройки дороги.

**Требуется** написать программу, которая вычислит число возможных планов постройки кольцевой автомобильной дороги с соблюдением указанных требований и найдет такой план, для которого длина дороги будет минимальной.

### Формат входных данных

Входной файл содержит четыре строки. Каждая из них содержит по два целых числа:  $x_i$  и  $y_i$  – координаты места расположения достопримечательности. Первая строка описывает первую достопримечательность, вторая – вторую, третья – третью, четвертая – четвертую. Никакие две достопримечательности не находятся в одной точке.

Все числа во входном файле не превосходят 100 по абсолютной величине.

### **Формат выходных данных**

В первой строке выходного файла требуется вывести число возможных планов постройки кольцевой автомобильной дороги. Если таких планов бесконечно много, необходимо вывести в первой строке выходного файла слово *Infinity*.

На второй строке требуется вывести координаты центра дороги минимальной длины и ее радиус. Если существует несколько разных способов построения дороги минимальной длины, необходимо вывести координаты центра и радиус любой из них. Координаты центра и радиус дороги должны быть выведены с точностью не хуже  $10^{-5}$ .

### **Пример входных и выходных данных**

<b>road.in</b>	<b>road.out</b>
0 0 0 1 1 0 2 2	7 1.5 0.5 1.14412281
0 0 0 1 1 0 1 1	Infinity 0.5 0.5 0.0

## 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  $m$  ( $1 \leq n, m \leq 100$ ) – высоту и ширину исходного изображения (в пикселях).

Следующие  $n$  строк содержат описание исходного изображения. Каждая строка состоит из  $m$  символов «В» и «W». Символ «В» 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 ( $2 \leq n \leq 1000$ ). Последующие  $n$  строк содержат записи Максима, причем каждая строка содержит две компоненты: вещественное число  $f_i$  – частоту, выставленную на тюнере, в герцах ( $30 \leq f_i \leq 4000$ ), 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  $n$  ( $1 \leq n \leq 100000$ ) – количество деталей двигателя. Вторая строка содержит  $n$  натуральных чисел  $p_1, p_2 \dots p_n$ , 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

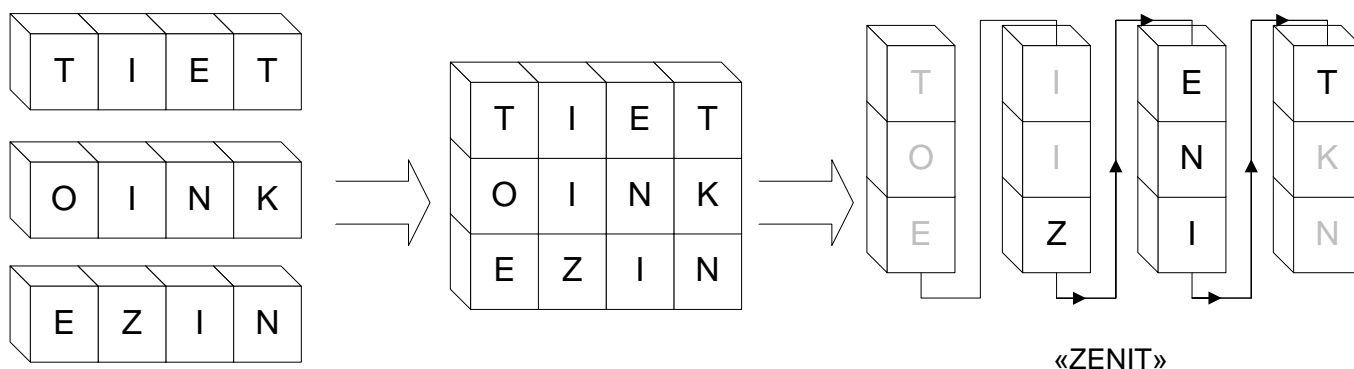
Имя входного файла:	word.in
Имя выходного файла:	word.out
Максимальное время работы на одном тесте:	2 секунды
Максимальный объем используемой памяти:	64 мегабайта
Максимальная оценка:	100 баллов

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  $1 \times 1 \times L$ , 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 прочесть 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: «TOEIIZENITKN».

**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 содержит два натуральных числа  $N$  и  $L$  – число различных типов блоков на складе и длина каждого блока соответственно ( $1 \leq N \leq 100$ ,  $1 \leq L \leq 100$ ). 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 ( $1 \leq |s| \leq 200$ ). 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**

<b>word.in</b>	<b>word.out</b>
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  $N \leq 5$ , будут оцениваться из 50 баллов.