Задача 1. Соревнование картингистов

Максимальное время работы на одном тесте: 2 секунды
Максимальный объем используемой памяти: 64 мегабайта

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, 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 ($1 \le n, m \le 100$). 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. Треугольник Максима

Максимальное время работы на одном тесте: 2 секунды
Максимальный объем используемой памяти: 64 мегабайта

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 3. Булева функция

Максимальное время работы на одном тесте: 2 секунд
Максимальный объем используемой памяти: 64 мегабайта

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 $a_1, a_2, ..., a_N$, 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, …, 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 OR 1) и 0, то есть, 1 и 0. После второго (и последнего) шага получается результат цепного вычисления, равный 1, так как 1 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* ($2 \le N \le 100\,000$).

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

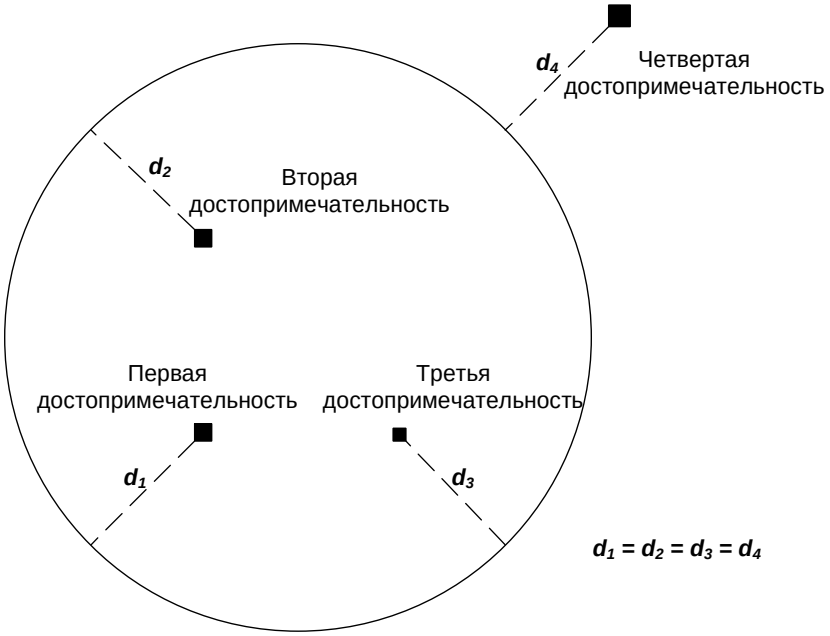
Решения, правильно работающие только при $N \leq 20$, будут оцениваться из 40 баллов.

Задача 4. Кольцевая автодорога

Максимальное время работы на одном тесте: 2 секунды
Максимальный объем используемой памяти: 64 мегабайта

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 построить дорогу минимальной длины, необходимо вывести любой из них. Координаты центра и радиус дороги должны быть выведены с точностью не хуже 10^{-5}.

Пример входных и выходных данных

road.in	road.out
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