Задача A. Мутация

Имя входного файла:	`mutation.in`
Имя выходного файла:	`mutation.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Рассмотрим следующий оператор мутации. Пусть $s \in \{0,1\}^n$ — битовая строка. В результате мутации каждый бит строки s инвертируется с вероятностью $1/n$. Для заданных пар строк s_i и t_i, $1 \le i \le m$ найдите вероятность получения t_i при применении рассмотренного оператора мутации к s_i.

Формат входного файла

В первой строке входного файла содержатся числа n и m ($1 \le n \le 10$, $1 \le m \le 100\,000$) — размер битовых строк и число пар строк, для которых нужно ответить на вопрос задачи. В каждых из последующих m пар строк входного файла записаны битовые строки s_i и t_i, $1 \le i \le m$ в виде последовательностей чисел 0 и 1 длины n.

Формат выходного файла

Для каждой пары строк s_i и t_i выведите в выходной файл строку, состоящую из одного вещественного числа — вероятности получения строки t_i из строки s_i в результате мутации. Ответ будет засчитан, если он отличается от правильного не более, чем на 10^{-9}.

Пример

mutation.in	mutation.out
5 2 11111 00000 11110 00000	0.00032 0.00128

Задача B. Кроссовер

Имя входного файла:	`crossover.in`
Имя выходного файла:	`crossover.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Пусть A — множество, состоящее из m битовых строк длины n, представляющих собой поколение генетического алгоритма. Необходимо для заданной битовой строки s (также размера n) выяснить, может ли она получиться из каких-либо двух строк из множества A в результате применения операторов одноточечного, двухточечного и однородного кроссовера.

Отметим, что при выполнении одноточечного и двухточечного кроссоверов точки скрещивания могут находиться до первого символа строки или после последнего символа, а в случае двухточечного кроссовера точки скрещивания могут совпадать.

Формат входного файла

В первой строке входного файла содержатся два числа m и n ($1 \le m, n \le 10^5$, $m \cdot n \le 10^6$) — число битовых строк в множестве A и их длина. В последующих m строках входного файла записаны последовательности длины n из нулей и единиц, задающие множество A. В последней строке входного файла записана битовая строка s в том же формате, что и строки из множества A.

Формат выходного файла

В первой строке выходного файла должно содержаться слово «`YES`», если строка s может получиться в результате применения одноточечного кроссовера к каким-либо двум (возможно, одинаковым) строкам из множества A, и слово «`NO`» в противном случае. Аналогично, во второй и третьей строках выходного файла должны содержаться слова «`YES`» или «`NO`» в зависимости от того, может ли получиться строка s в результате двухточечного и однородного кроссовера соответственно.

Пример

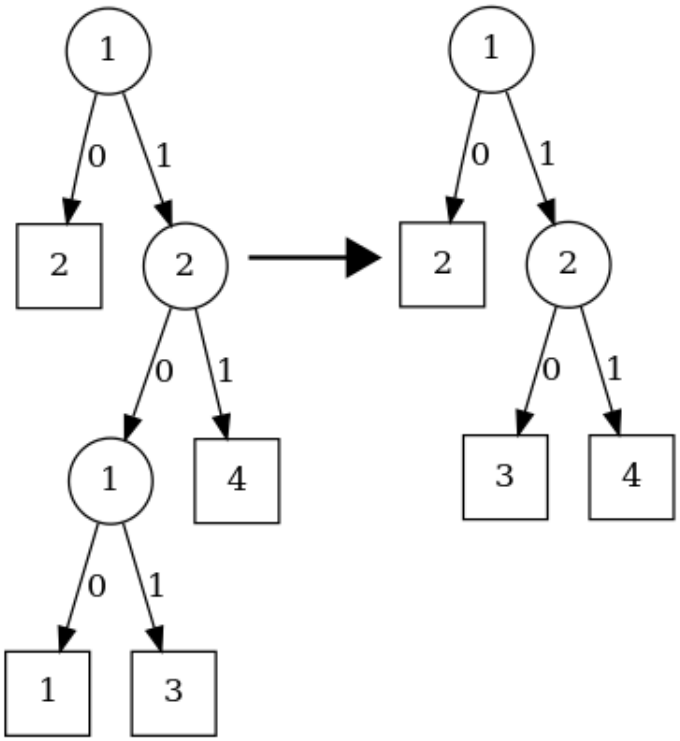
crossover.in	crossover.out
2 4 0000 1111 0101	NO NO YES

Задача C. Деревья решений

Имя входного файла:	`trees.in`
Имя выходного файла:	`trees.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Деревья решений — один из способов задания функции нескольких аргументов, каждый из которых пробегает конечное множество значений. В частности, деревья решений могут использоваться для представления функции переходов автомата. Рассмотрим дерево решений, соответствующее функции переходов некоторого состояния автомата. Будем считать, что переходы из состояния в состояние осуществляются на основе значений m предикатов. В этом случае каждый лист дерева решений помечен значением функции переходов, а каждая вершина ветвления — номером предиката, в зависимости от значения которого при вычислении функции переходов будет выбрана одна из дочерних вершин.

В случае, если построение деревьев решений осуществляется на основе эволюционных вычислений, возникает задача удаления недостижимых ветвей дерева. Вам предстоит решить эту задачу. Более формально, если вершина ветвления v помечена предикатом с номером p и если среди ее предков есть другая вершина u, помеченная предикатом с тем же номером p, то вершина v должна быть удалена вместе с ее недостижимым поддеревом. Оставшееся поддерево вершины v должно быть присоединено к бывшему родителю v. Пример указанного преобразования приведен на рисунке.



Формат входного файла

В первой строке входного файла содержится число k ($1 \le k \le 2 \cdot 10^5$) — число вершин дерева решений. В последующих k строках входного файла последовательно с первой по k-ю описаны вершины дерева. Формат задания листа: `leaf` a_i, где a_i ($1 \le a_i \le 10^9$) — значение

функции переходов для i-й вершины. Формат задания вершины ветвления: `choice` p_i f_i t_i, где p_i ($1 \le p_i \le 10^9$) — номер предиката для i-й вершины, f_i и t_i — номера дочерних вершин, соответствующих переходам по 0 и 1. Корень дерева является вершиной с номером 1.

Формат выходного файла

Запишите в выходной файл дерево решений, получившееся из исходного в результате удаления всех недостижимых ветвей. Используйте тот же формат, что и во входном файле.

Пример

trees.in	trees.out
7 choice 1 2 3 leaf 2 choice 2 4 5 choice 1 6 7 leaf 4 leaf 1 leaf 3	5 choice 1 2 3 leaf 2 choice 2 4 5 leaf 3 leaf 4

Приведенный пример соответствует рисунку.

Задача D. Стартовое состояние

Имя входного файла:	`start.in`
Имя выходного файла:	`start.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Задан детерминированный управляющий автомат Мура. Пусть множество входных событий для него равно $\{0, 1\}$, а множество выходных воздействий — $\{a, b, \ldots, z\}$. При выполнении перехода автомат вырабатывает выходное воздействие, которым помечено то состояние, куда был совершен переход.

Задана также последовательность выходных воздействий $z = a_1 \ldots a_m$ длины m. Известно, что последовательность z была получена в результате работы автомата на некоторых входных данных. Требуется найти номера состояний, из которых существует последовательность переходов, результатом выполнения которой будет выработка последовательности z.

Формат входного файла

В первой строке входного файла содержатся два числа m и n ($1 \le m \le 5 \cdot 10^5$, $1 \le n \le 500$) — длина последовательности z и число состояний автомата. В следующих n строках последовательно от первого до n-го описаны состояния автомата Мура. Каждая строка имеет формат $t_{i,0}\ t_{i,1}\ b_i$, где $t_{i,0}$ и $t_{i,1}$ — номера состояний, куда ведут переходы из состояния i по событиям 0 и 1 соответственно, b_i — выходное воздействие, которым помечено состояние i. В последней строке входного файла записана последовательность выходных воздействий z.

Формат выходного файла

В единственной строке выходного файла выведите число состояний, являющихся решением задачи, а затем через пробел перечислите в порядке возрастания номера этих состояний.

Пример

start.in	start.out
2 2 2 2 a 2 1 b ab	1 2

Задача E. Дискретные выходные воздействия

Имя входного файла:	`discrete.in`
Имя выходного файла:	`discrete.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Рассмотрим задачу построения управляющих автоматов с дискретными выходными воздействиями по обучающим примерам, или тестам. i-й ($i = 1 \dots m$) тест в обучающем наборе представляет собой две последовательности длины $\text{len}[i]$: последовательность входных событий $\text{in}[i] \in \{e_0, e_1\}^{\text{len}[i]}$ и последовательность дискретных выходных воздействий $\text{out}[i] \in \{a, b, \dots, z\}^{\text{len}[i]}$.

Задача управляющего автомата — в ответ на последовательности событий $\text{in}[i]$ выдавать последовательности $\text{ans}[i]$, в определенном смысле близкие к последовательностям $\text{out}[i]$. Близость поведения автомата к поведению, показанному в тестах, выражается функцией приспособленности:

$$f = -\sum_{i=1}^{m} \frac{1}{\text{len}[i]} \sum_{j=1}^{\text{len}[i]} [\text{out}[i][j] \neq \text{ans}[i][j]] \,.$$

Последовательности $\text{ans}[i]$ формируются следующим образом. Перед обработкой i-го теста автомат находится в начальном состоянии. Далее ему последовательно подаются на вход события из последовательности $\text{in}[i]$. Значение $\text{ans}[i][t]$ равно выходному воздействию, сопоставленному переходу автомата, который выполняется в момент времени t.

Каркасом управляющего автомата Мили будем называть частично заданный автомат, переходам которого не сопоставлены выходные воздействия. Необходимо расставить выходные воздействия на переходах заданного каркаса автомата так, чтобы максимизировать значение функции приспособленности f.

Формат входного файла

В первой строке входного файла содержатся числа n и m ($1 \le n \le 10$, $1 \le m \le 2000$) — число состояний управляющего автомата и число тестов в обучающем наборе.

В следующих n строках последовательно перечисляются состояния каркаса автомата от первого до n-го. Каждая строка имеет вид $t_{i,0}\ t_{i,1}$, где $t_{i,0}$ и $t_{i,1}$ — номера состояний, в которые осуществляется переход из i-го состояния по событиям e_0 и e_1 соответственно. Первое состояние является начальным.

Далее, в последующих m строках последовательно перечислены тесты в формате $\text{len}[i]$ $\text{in}[i]$ $\text{out}[i]$, где i — номер теста, $1 \le \text{len}[i] \le 10^4$.

Формат выходного файла

В выходной файл выведите n строк. Строка i должна иметь формат $z_{i,0}\ z_{i,1}$, где $z_{i,0}$ и $z_{i,1}$ — оптимальные дискретные выходные воздействия на переходах из состояния i по событиями e_0 и e_1 соответственно. Если ответ не единственный, выведите любой.

Пример

discrete.in	discrete.out
2 2 2 1 1 2 7 0110001 accbabc 4 1111 cccc	a c b c

Задача F. Непрерывные выходные воздействия

Имя входного файла:	`continuous.in`
Имя выходного файла:	`continuous.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Рассмотрим задачу построения управляющих автоматов с непрерывными выходными воздействиями по обучающим примерам, или тестам. i-й ($i = 1 \dots m$) тест в обучающем наборе представляет собой две последовательности длины $\text{len}[i]$: последовательность входных событий $\text{in}[i] \in \{e_0, e_1\}^{\text{len}[i]}$ и последовательность непрерывных выходных воздействий $\text{out}[i] \in \mathbb{R}^{\text{len}[i]}$.

Задача управляющего автомата — в ответ на последовательности событий $\text{in}[i]$ выдавать последовательности $\text{ans}[i]$, в определенном смысле близкие к последовательностям $\text{out}[i]$. Близость поведения автомата к поведению, показанному в тестах, выражается функцией приспособленности:

$$f = -\sum_{i=1}^{m} \frac{1}{\text{len}[i]} \sum_{j=1}^{\text{len}[i]} \left(\text{out}[i][j] - \text{ans}[i][j]\right)^2.$$

Последовательности $\text{ans}[i]$ формируются следующим образом. Перед обработкой i-го теста автомат находится в начальном состоянии. Далее ему последовательно подаются на вход события из последовательности $\text{in}[i]$. Значение $\text{ans}[i][1]$ равно выходному воздействию, сопоставленному переходу автомата, который выполняется в момент времени времени $t = 1$. Далее, если в момент времени $t, t > 1$ выполняется переход с выходным воздействием Δu, то $\text{ans}[i][t] = \text{ans}[i][t-1] + \Delta u$.

Каркасом управляющего автомата Мили будем называть частично заданный автомат, переходам которого не сопоставлены выходные воздействия. Необходимо расставить выходные воздействия на переходах заданного каркаса автомата так, чтобы максимизировать значение функции приспособленности f.

Формат входного файла

В первой строке входного файла содержится числа n и m ($1 \le n \le 10$, $1 \le m \le 50$) — число состояний управляющего автомата и число тестов в обучающем наборе.

В следующих n строках последовательно перечисляются состояния каркаса автомата от первого до n-го. Каждая строка имеет вид $t_{i,0}\ t_{i,1}$, где $t_{i,0}$ и $t_{i,1}$ — номера состояний, в которые осуществляется переход из i-го состояния по событиям e_0 и e_1 соответственно. Первое состояние является начальным.

Далее, в последующих m строках последовательно перечислены тесты в формате $\text{len}[i]$ $\text{in}[i]$ $\text{out}[i]$, где i — номер теста, $1 \le \text{len}[i] \le 250$. Последовательности $\text{in}[i]$ — это **строки** длиной $\text{len}[i]$, состоящие из нулей (событие e_0) и единиц (событие e_1). Последовательностях $\text{out}[i]$ состоят из $\text{len}[i]$ вещественных чисел, записанных через пробел и имеющих до шести знаков после запятой.

Формат выходного файла

В выходной файл выведите n строк. Строка i должна иметь формат $z_{i,0}\ z_{i,1}$, где $z_{i,0}$ и $z_{i,1}$ — вещественные числа, равные оптимальным выходным воздействиям на переходах из состояния i по событиями e_0 и e_1 соответственно. Если от выходного воздействия на некотором переходе значение функции приспособленности не зависит, будем считать, что оптимальное

значение для этого воздействия равно нулю. Ответ засчитывается, если каждое выведенное выходное воздействие отличается от оптимального не более чем на 10^{-6}.

Пример

continuous.in	continuous.out
2 2 1 2 1 2 1 0 0.125032 1 0 0.126637	0.125835 0 0 0

Задача G. Умный муравей

Имя входного файла:	`artificial.in`
Имя выходного файла:	`artificial.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Задано тороидальное поле, состоящее из клеток и имеющее размер $m \times m$. В части клеток поля расположена еда.

По полю может перемещаться муравей. Положение муравья задается клеткой, в которой он находится. Изначально муравей находится в левой верхней клетке поля и смотрит вправо. Муравей может видеть, есть ли еда в клетке перед ним, и на каждом шаге должен выполнить ровно одно из трех действий:

- повернуться на 90 градусов налево;
- повернуться на 90 градусов направо;
- переместиться на клетку вперед и съесть еду, если она есть в клетке.

Задача муравья — съесть всю еду, находящуюся на поле, за k шагов. Требуется построить автомат Мили, под управлением которого муравей решит свою задачу. Автомат должен иметь n состояний. На каждом такте автомат получает на вход события e_0 или e_1 в зависимости от отсутствия или наличия еды в клетке перед муравьем. Переход автомата может быть помечен одним из трех выходных воздействий: L, R и M, обозначающих поворот налево, поворот направо и перемещение вперед соответственно. Гарантируется, что искомый автомат существует.

Формат входного файла

В первой и единственной строке входного файла содержится единственное число — порядковый номер теста. Все тесты к данной задаче приведены далее в условии.

Формат описания теста таков. В первой строке теста содержатся числа m, n и k. Далее записаны m строк по m символов в каждой, описывающие поле. Звездочка соответствует наличию еды, точка — отсутствию. Гарантируется, что в левой верхней клетке поля нет еды.

Формат выходного файла

В i-й строке выходного файла опишите i-е состояние автомата, решающего задачу, в формате $t_{i,0}\ t_{i,1}\ a_{i,0}\ a_{i,1}$, где $t_{i,0}$ и $t_{i,1}$ — номера состояний на переходах по e_0 и e_1 (состояния нумеруются от 1 до n), $a_{i,0}$ и $a_{i,1}$ — выходные воздействия на этих переходах.

Пример

artificial.in	artificial.out
1	2 1 R M 3 3 M M 3 4 R M 2 2 L M

Номер теста	Тест
1	10 4 22 .*..... .*..... .*****.... .....*.... .....*.... ....*.... .....*.... .****..... .....
2	10 1 117 .********* ********** ********** ********** ********** ********** ********** ********** ********** **********
3	10 3 117 .*.*.*.*.* *.*.*.*.*. .*.*.*.*.* *.*.*.*.*. .*.*.*.*.* *.*.*.*.*. .*.*.*.*.* *.*.*.*.*. .*.*.*.*.* *.*.*.*.*.
4	10 4 22 .**..... ....*..... ....*..... ....**.. ....* .....* .....

Номер теста	Тест
5	10 3 47 .**....... ....*..... ....*..... ....**.. ....* ....* ....
6	10 5 48 .***...... ....*..... ....*...** *..*...*.. *.*....... *.*....*.. *.*....*.. *..*.**... *......... .**.**.*..
7	10 4 52 .***...... ....*..... ....*...** *..*...*.. *.*....... *.*....*.. *.*....*.. *..*.**... *......... .**.**.*..
8	10 5 58 .***...... ....*..... ....*...** *..*...*.. *.*....... *.*....*.. *.*....... *..*.**.*. *......... .**.**.*..

Номер теста	Тест
9	20 6 266 .****.....*..... ....*.....*..... ....*.....*..... *****.....*.....** ......***.....*.. ..****.....*.. **.....** ....**.**..**....*... .*.....*... .*....*.*.*.....*... .....*.....*.....*.. .*...*.....*.....*. .*...*....*.....*.. .*..*.....*.....*... ..**.....*.....*.. .....*.....*. .....*....*.. .....*..... .....*.....
10	20 6 283 .***.....**.***... ...*..***..... ...***...**..*...*** .....* *.....* **..... ..*.....**....**.... ..*...**....**..*... ....**....**....*... .....*... ..********.....*... .*....****.....*... .*...*.....*... .*...*.....*... .*....**********.... .*..... .*.....***..... ..*****...***..... .....***.... .....

Задача H. Интерактивная минимизация

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

На единичном гиперкубе $[0,1]^n$ определена функция f, которая задается формулой:

$$f(x_1, ..., x_n) = \sum_{i=1}^{n} q_i (x_i - a_i) + q_0 \sum_{j=1}^{n} \cos(2\pi t_j (x_j - a_j)),$$

где $q_0, ..., q_n \in [-1, 1]$, $a_1, ..., a_n \in [0, 1]$, $t_1, ..., t_n \in [1, 10]$. Требуется найти глобальный минимум функции f. Значения параметров $q_0, ..., q_n$, $a_1, ..., a_n$ и $t_1, ..., t_n$ неизвестны: чтобы узнать значение $f(x_1, ..., x_n)$, необходимо сделать запрос проверяющей программе.

Интерактивный протокол

Ваша программа может делать два типа запросов. **Во-первых**, программа может запросить значение f в точке $(x_1, ..., x_n)$. После этого программа может прочитать ответ на запрос из стандартного входа. Число запросов данного типа не должно превышать $1000n$. **Во-вторых**, программа может дать ответ на задачу, указав минимум функции f. После такого запроса программа должна прекратить свое взаимодействие с проверяющей программой.

В конце каждого запроса необходимо вывести перевод строки и очистить буфер потока стандартного выхода командой flush.

Формат входного файла

Первая строка стандартного входа содержит число n ($1 \le n \le 20$) — размерность области определения функции f. Последующие строки стандартного входа содержат ответы на запросы вашей программы. Каждая строка входа состоит из единственного вещественного числа — значения f в желаемой точке. Все значения даны с девятью знаками после запятой.

Формат выходного файла

При запросе значения функции f строка выходного потока должна состоять из n вещественных чисел от 0 до 1, разделенных пробелом. При запросе с указанием минимума следует вывести слово «minimum», после чего через пробел записать ответ на задачу — одно вещественное число. Ответ будет засчитан, если он отличается от глобального минимума функции не более чем на 10^{-3}.

Пример

standard input	standard output
2 0.120000000 0.110000000 0.090000000 0.080000000	0.25 0.25 0.35 0.35 0.45 0.45 0.55 0.55 minimum 0.08

Приведенный пример лишь показывает протокол в действии. При таком небольшом числе запросов трудно определить минимум функции.

Задача I. Пять минимумов

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`
Ограничение по времени:	50 секунд
Ограничение по памяти:	256 мегабайт

В данной задаче Вам предлагается минимизировать вещественнозначную функцию. Точная формула этой функции не будет Вам известно, но она всегда будет иметь следующий вид:

$$F\left(x_1, \ldots, x_N\right) = \min_{i=1\ldots5} \left(\beta_i + \gamma_i \sum_{j=1}^{N} \left(x_j - \alpha_{j,i}\right)^2 \right),$$

где N — размерность задачи ($1 \le N \le 10$), а ограничения на параметры задачи таковы: $|\alpha_{j,i}| \le 10$, $|\beta_i| \le 10$, $1 \le \gamma_i \le 2$. Говоря упрощенно, функция является минимумом по пяти N-мерным параболам, которые обладают разными смещениями оптимума и разными масштабными коэффициентами. Никакие два центра парабол (то есть, локальные минимумы функции) не находятся друг к другу ближе, чем на расстояние 1.0.

Единственный способ получить какую-либо информацию о функции — делать запросы вида $(x_1, \ldots, x_N)$, где x_i — вещественные числа, не превосходящие 10 по модулю. Тестирующая система будет возвращать значения F на Ваших запросах. Для осуществления оптимизации у Вас будет возможность сделать $10^4 \times N^2$ запросов.

Чтобы усложнить жизнь, ответы на Ваши запросы будут немного испорчены. Более формально, тестирующая система будет добавлять случайное значение из интервала $[-0.5; 0.5]$ к значению функции перед тем, как отправить его Вашему решению.

Чтобы упростить жизнь, Вы не должны искать точный глобальный минимум. Вместо этого, будет считаться, что Вы решили задачу, если Ваше решение нашло точку x, такую что $F(x)$ не больше, чем *второй по порядку локальный минимум*, с некоторой *точностью* ε.

Чтобы Вам было проще оценивать качество Ваших решений, тесты будут обладать достаточно регулярной структурой. Всего будет 50 тестов, пронумерованных от 1 до 50, которые представляют 10 различных функций. Для теста с номером T выполняется следующее:

- $1 + (T - 1) \bmod 10$ — номер функции, который совпадает с размерностью задачи N;
- $\lfloor (T-1)/10 \rfloor$ определяет точность: $\varepsilon = 10^{2-\lfloor (T-1)/10 \rfloor}$.

Например, тест 28 содержит восьмую, восьмимерную, функцию, и она будет считаться решенной, если решение найдет точку, не более чем на 1.0 превосходящую второй локальный минимум.

Для удобства тестирования, приведем первую из функций:

$$F_1(x) = \min\left(1 + 2(x-1)^2, 2 + 1.8(x-2)^2, 3 + 1.6(x-3)^2, 4 + 1.4(x-4)^2, 5 + 1.2(x-5)^2\right).$$

Интерактивный протокол

Вначале, тестирующая система передает Вашему решению размерность задачи N. Затем, решение делает запросы, а тестирующая система отвечает на эти запросы (внося случайный

шум, как описано выше). Когда решение делает запрос, такой что $F(x) \leq F(y_2) + \varepsilon$ (где y_2 — это второй по порядку локальный минимум, а ε — точность, как описано выше), тестирующая система ответит строкой «Bingo» и отключится от Вашего решения. Сделайте так, чтобы Ваше решение корректно завершало работу, когда получит в качестве ответа указанную строку.

Ограничение на число запросов составляет $10^4 \times N^2$. Если Ваше решение сделает $(10^4 \times N^2 + 1)$-ый запрос, тестирующая система прекратит работу, и Вы получите вердикт «Wrong Answer». Также Вы получите такой результат, если Ваше решение слишком рано прервет работу.

Если Ваше решение использует слишком много ресурсов процессора (больше, чем ограничение на время работы), Вы получите вердикт «Time Limit Exceeded». **Не думайте, что 50 секунд — это очень много** — Ваше решение будет большую часть этого времени делать ввод и вывод. Если Ваше решение завершит работу некорректно (с ошибкой времени выполнения), Вы получите вердикт «Runtime Error».

Если Ваш запрос содержит что-то, кроме N вещественных чисел, не превышающих 10 по модулю, тестирующая система прекратит работу, и Вы получите вердикт «Presentation Error».

Наконец, если все хорошо (Ваше решение нашло достаточно хороший набор аргументов за достаточно небольшое число запросов) на каждом тесте, Вы получите вердикт «Accepted», что означает, что задача решена.

Формат входного файла

В первой строке находится число N, размерность задачи. В последующих строках находятся ответы на запросы. Каждый ответ — это либо вещественное число с не более чем шестью знаками после точки, или строка «Bingo» (без кавычек). В случае получения строки «Bingo» Ваше решение должно прекратить работу.

Формат выходного файла

Выводите запросы по одному на строке. Каждый запрос должен состоять из ровно N вещественных чисел, соответствующих $x_1, \ldots, x_N$, разделенных пробелами, такими, что ни одно из них не превосходит 10 по модулю.

Пример

standard input	standard output
1 2.997112 2.169921 Bingo	0 2 1