Об уровнях олимпиады

В этой интернет-олимпиаде мы продолжаем эксперимент с уровнями сложности.

Вы можете самостоятельно выбрать, задачи какого уровня решать.

Если вы хотите участвовать в олимпиаде базового уровня, решайте задачи A, B, C и D.

Если вы хотите участвовать в олимпиаде усложненного уровня, решайте задачи C, D, E и F.

У базового и усложненного уровней будут отдельные таблицы результатов.

Если вы отправите решение задачи E или F, вы автоматически будете классифицированы в таблице усложненного уровня (даже если ваше решение не пройдет тесты из примера). Иначе вы будете классифицированы в таблице базового уровня.

Задача A. Футбольный чемпионат

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

Вася — ярый футбольный фанат. Недавно он узнал, что матч с участием его любимой команды будет последним в чемпионате, и результаты всех остальных матчей будут известны до его окончания.

Естественно, Вася хочет рассчитать, на каком месте в турнирной таблице окажется его команда в зависимости от результата матча. Несмотря на то, что задача не так сложна, Вася боится не справиться с волнением, поэтому попросил вас написать программу, вычисляющую это за него.

Имейте в виду, что победившая команда получает 3 очка, а проигравшая — 0. В случае ничейного исхода матча, обе команды получают по одному очку.

Формат входного файла

Первая строка входного файла содержит натуральное число n — количество команд, участвующих в чемпионате ($1 \le n \le 20$). Вторая строка входного файла содержит название команды, за которую болеет Вася, а третья — название второго участника последнего матча. Далее следует описание турнирной таблицы к окончанию всех матчей, кроме последнего. Каждая из последующих n строк содержит название команды и количество очков. Все названия состоят из заглавных латинских букв и их длина не превышает двадцати символов.

Формат выходного файла

В выходной файл выведите три числа разделенные пробелом — положение команды, за которую болеет Вася в случае ее победы в последнем матче, ничьей и поражении соответственно. Учтите, что при равенстве очков более высокое место имеет команда с лексикографически меньшим названием.

Примеры

football.in	football.out
8 ZENIT SATURN SATURN 45 SPARTAKM 59 LOKOMOTIV 41 CSKA 53 MOSKVA 52 ZENIT 58 DINAMO 41 AMKAR 41	1 2 2
8 RUBIN ZENIT MOSKVA 48 ZENIT 54 RUBIN 63 SATURN 45 CSKA 52 DINAMO 42 LOKOMOTIV 54 SPARTAKM 55	1 1 1

Задача B. Строка

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

Вася очень любит строки, а Петя — числа. Но оба они любят последовательности. Поэтому Вася написал на доске строку s, а Петя — последовательность из n натуральных чисел $a_1, a_2 \ldots a_n$.

Теперь ребят интересует, есть ли в строке s такая подпоследовательность символов c_k, что первые a_1 символов в ней равны между собой, символы с $(a_1 + 1)$-го по $(a_1 + a_2)$ — тоже совпадают и так далее. То есть для каждого i $(1 \le i \le n)$ символы c_k при $\sum_{j=1}^{i-1} a_j + 1 \le k \le \sum_{j=1}^{i} a_j$ равны между собой.

Если же подпоследовательности, обладающей таким свойством, в строке s не существует, ребят интересует наименьшее количество символов, которые достаточно дописать в конец строки s, чтобы указанное свойство выполнялось.

Формат входного файла

В первой строке входного файла одно натуральное число n $(1 \le n \le 1000)$. Во второй строке содержится n натуральных чисел разделенных пробелом — a_i $(\sum_{i=1}^{n} a_i \le 1000)$. Строка s непуста и состоит не более чем из 1000 строчных латинских букв.

Формат выходного файла

Если искомая подпоследовательность существует, то в выходной файл требуется вывести «`0`». Иначе необходимо вывести количество символов, которое достаточно дописать в конец строки s для выполнения свойства.

Примеры

string.in	string.out
3 1 1 2 abacaba	0
3 1 2 2 nosolution	1

Задача C. Квадратный корень

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

Аня и Борис учатся в одном классе. Совсем недавно на уроке математики они узнали, что такое квадратный корень. Это математическое понятие показалось им достаточно сложным, и они решили закрепить его.

Для этого они придумали игру: Аня загадывает натуральное число A, а Борис — натуральное число B, после чего они вместе смотрят, что больше — $\sqrt{A}$ или $\sqrt{B}$?

После нескольких попыток они поняли, что игра получилась слишком простой. Действительно, $\sqrt{A} > \sqrt{B}$ тогда и только тогда, когда $A > B$. Поэтому они решили усложнить игру.

Теперь Аня загадывает два неотрицательных целых числа A, B, а Борис загадывает два неотрицательных целых числа C, D. Необходимо узнать, что больше — $A + \sqrt{B}$ или $C + \sqrt{D}$?

Новая игра получилась гораздо сложнее предыдущей. Через некоторое время дети попросили Вас проверить несколько особенно сложных случаев, которые им встретились. Помогите им и напишите программу, которая определяет, что больше — $A + \sqrt{B}$ или $C + \sqrt{D}$?

Формат входного файла

В первой строке входного файла находится число N ($1 \le N \le 10000$) — число случаев, которые необходимо рассмотреть. В каждой из последующих N строк находится по четыре целых числа A_i, B_i, C_i, D_i — описание случая. Известно, что $0 \le A_i, B_i, C_i, D_i \le 10^9$.

Формат выходного файла

Для каждого случая вывести в выходной файл одну строку. Пусть $L_i = A_i + \sqrt{B_i}$, а $R_i = C_i + \sqrt{D_i}$. В строке должно содержаться слово «`Less`», если $L_i < R_i$, «`Equal`», если $L_i = R_i$, и «`Greater`», если $L_i > R_i$.

Примеры

sqroot.in	sqroot.out
3 1 2 2 3 0 1 1 0 3 2 1 0	Less Equal Greater

Задача D. Системы вложенных коллайдеров

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

Правительству страны Флатландия понравилось строить адронные коллайдеры. Поэтому было решено построить систему вложенных коллайдеров.

Каждый коллайдер — это окружность с центром в точке с целыми координатами и целым положительным радиусом. Система вложенных коллайдеров состоит из одного или более таких коллайдеров. При этом, каждый коллайдер, кроме самого большого, должен находиться внутри предыдущего, но может иметь с ним одну общую точку.

Для постройки системы вложенных коллайдеров выделена прямоугольная площадка, размеченная прямоугольной системой координат так, что противоположные углы площадки имеют координаты $(-n, -m)$ и (n, m). Конечно же, никакой коллайдер не должен выходить за пределы этой площадки, но может иметь с ее границей общие точки.

Правительству Флатландии стало интересно, сколькими способами они могут построить такую систему вложенных коллайдеров, и они поручили эту задачи вам. Так как это число может быть большим, достаточно посчитать его по модулю 10^9.

Формат входного файла

Во входном файле заданы два целых числа n и m ($1 \le n, m \le 1000$).

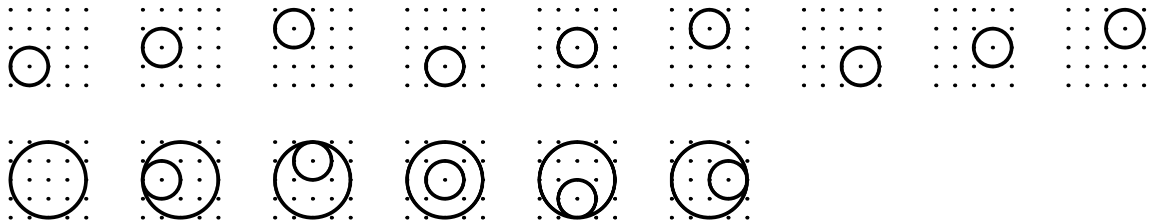
Формат выходного файла

В выходной файл выведите остаток от деления числа способов построить систему вложенных коллайдеров на 10^9.

Примеры

colliders.in	colliders.out
1 3	5
2 2	15

Иллюстрация ко второму примеру:



Задача E. Три цвета

Имя входного файла:	`3colors.in`
Имя выходного файла:	`3colors.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

Число три в информатике играет ключевую роль. Часто некоторые задачи легко решаются, если некоторый параметр в задаче равен двум или одному, но становятся сложными, если этот параметр равен трем. Например, задача о раскраске графа в два цвета — это проверка на двудольность, а раскраска в три цвета — трудная задача.

Мы решили, что это нечестно и придумали задачу, которая легко решается для трех цветов, но гораздо труднее для двух.

Рассмотрим двудольный граф G, разрешим раскрашивать его ребра в три цвета: 0, 1 и 2. Будем обозначать сумму цветов ребер, один из концов которых равен u, как $s(u)$. Будем называть такую раскраску *различающей соседей*, если для любых двух вершин u и v, соединенных ребром, выполнено неравенство $s(u) \neq s(v)$.

По заданному двудольному графу G найдите его раскраску в три цвета, различающую соседей.

Формат входного файла

Первая строка входного файла содержит три целых числа n_1, n_2 и m — количество вершин в каждой из долей и количество ребер в графе, соответственно ($1 \le n_1, n_2 \le 1500$, $1 \le m \le 10000$).

Следующие m строк описывают ребра, каждое ребро описывается двумя целыми числами — номерами вершин, которые оно соединяет. Вершины в каждой доле независимо нумеруются, начиная с единицы.

Формат выходного файла

Если решения не существует, выведите «−1».

Иначе выведите m целых чисел — цвета ребер. Выводите цвета ребер в порядке, в котором ребра заданы во входном файле.

Примеры

3colors.in	3colors.out
3 3 7 1 1 1 2 1 3 2 2 3 1 3 2 3 3	0 0 0 1 2 2 2

Задача F. Жестокие игры

Имя входного файла:	`shooting.in`
Имя выходного файла:	`shooting.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Алиса и Боб играют в жестокую игру на плоскости.

Игра протекает следующим образом. На плоскости расположено n препятствий, каждое из которых представляет собой отрезок. Сначала Алиса выбирает некоторую точку плоскости, не принадлежащую никакой прямой, содержащей препятствие, и встает в ней. Затем Боб выбирает некоторую точку и встает там. После этого Алиса стреляет в Боба из пистолета.

Разумеется, Боб хочет оказаться в безопасности. Поэтому он всегда выбирает такую точку, чтобы как можно больше препятствий было между ним и Алисой. Если конец препятствия оказывается на отрезке, соединяющем Алису с Бобом, то считается, что Боб спрятался за этим препятствием.

Алиса очень хотела бы пристрелить Боба, поэтому она старается выбрать такую точку, чтобы как можно меньше препятствий оказалось между ней и Бобом. Помогите ей это сделать.

Формат входного файла

Первая строка входного файла содержит число n — количество препятствий ($1 \le n \le 8$). Следующие n строк содержат по четыре целых числа x_1, y_1, x_2, y_2 координаты концов соответствующего препятствия. У препятствий нет общих точек. Координаты препятствий не превышают 100 по модулю.

Формат выходного файла

Первая строка выходного файла должна содержать k — минимальное количество препятствий, которое может оказаться между Алисой и Бобом, если они оба действуют оптимально. На второй строке выведите точку, в которую должна встать Алиса. Точка не должна принадлежать никакой прямой, содержащей препятствие.

Примеры

shooting.in	shooting.out
2 0 0 2 0 0 2 2 2	1 1.000000000000000 1.000000000000000