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. Ноутбук

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

> Не было печали — апдейтов накачали.
>
> Интернет-фольклор

Недавно Вова купил себе новый портативный компьютер (ноутбук). Преимущество такого компьютера состоит в том, что его можно носить с собой и работать на нем вдали от источников электричества, используя заряд аккумулятора.

После покупки Вова поставил на ноутбук свою любимую операционную систему *winBSD*, и в качестве эксперимента не выключал компьютер в течение нескольких недель.

За это время Вова определил, что его ноутбук при работе от аккумулятора разряжается полностью за x минут непрерывной работы. При этом, за единицу времени работы ноутбука, аккумулятор всегда теряет одно и то же количество энергии, независимо от того, какие задачи на нем выполняются.

Также Вова выяснил, что при работе от электрической сети аккумулятор ноутбука заряжается полностью за y минут. При этом, за единицу времени, аккумулятор всегда заряжается на одно и то же количество энергии. Разумеется, аккумулятор ноутбука не может быть заряжен более, чем на 100%.

Вова много перемещается по городу, поэтому его ноутбук n раз за сутки меняет источник энергии (с сети на аккумулятор и наоборот). Несмотря на это Вова никогда не выключает ноутбук.

Разумеется, при полной разрядке аккумулятора ноутбук выключается. Если же ноутбук с полностью разряженным аккумулятором подключить к сети, то он сразу включится.

После очередного обновления *winBSD* у Вовы перестал работать индикатор текущего заряда батареи. Однако, Вова не растерялся и написал программу, которая по расписанию смен источника энергии ноутбука определяет заряд батареи в конце суток.

Напишите и Вы такую программу. При этом можно считать, что в начале суток (ноль часов, ноль минут) ноутбук работает от сети и его аккумулятор полностью заряжен.

Формат входного файла

Первая строка входного файла содержит числа n, x, y ($0 \le n \le 20$, $1 \le x, y \le 1500$) — количество смен источника энергии ноутбука, время разряда и заряда батареи соответственно. Каждая из следующих n строк содержит время суток в формате «`hh:mm`», в которое была произведена смена источника питания. Во входном файле времена суток лежат в промежутке от `00:01` до `23:58`, перечислены в хронологическом порядке и попарно не совпадают.

Формат выходного файла

В выходной файл выведите процент заряда батареи в `23:59` с точностью не менее трех знаков после запятой.

Примеры

notebook.in	notebook.out
1 200 120 00:01	0
2 200 120 00:01 22:59	50.0
7 300 317 01:43 03:54 05:29 08:29 14:48 16:47 23:08	83.0

Задача B. Посты охраны

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

Для охраны одного сверхсекретного объекта необходимо организовать три поста охраны и огородить его территорию. Ограждение должно иметь форму окружности, так как это позволяет наиболее рационально использовать территорию. Вся область, находящаяся внутри окружности или на ней, называется *территорией объекта*.

Посты охраны, разумеется, должны находиться на территории объекта. По нормам безопасности расстояние между первым и вторым постом охраны должно быть не меньше a метров, между вторым и третьим — не меньше b метров, между третьим и первым — не меньше c метров.

Для того чтобы секретный объект был менее заметным, радиус ограждения должен быть как можно меньшим. Ваша задача состоит в том, чтобы написать программу, которая по числам a, b и c, определит наименьший радиус ограждения, удовлетворяющий описанным ограничениям.

Формат входного файла

Входной файл содержит три целых числа: a, b и c ($1 \le a, b, c \le 10^5$).

Формат выходного файла

В выходной файл выведите наименьший радиус ограждения (в метрах). Ответ будет засчитан, если он отличается от правильного не более, чем на 10^{-5}.

Примеры

guard.in	guard.out
6 10 8	5
10 10 10	5.77350269189625764509

Задача C. Фигурное катание

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

На одном соревновании по фигурному катанию в 2014 году решили ввести новую систему оценки выступлений фигуристов. Каждый из n элементов короткой программы было решено оценивать целым числом баллов от 0 до a_i, где a_i — максимальное число баллов, которое можно получить за i-ый элемент.

Общая оценка за выступление вычисляется как сумма баллов за каждый из элементов. При этом если фигурист получает за один из элементов (например, i-ый) менее b_i баллов, то все выступление оценивается в 0 баллов.

К сожалению, члены жюри не привыкли оценивать выступление по этим правилам, поэтому они выставляют одно число m — суммарную оценку. Перед ответственным секретарем соревнований Яном стоит задача привести оценки жюри к требуемому виду, то есть получить набор чисел $c_1, \ldots, c_n$, таких, что они удовлетворяют описанным ограничениям и оценка всего выступления равна m, или сообщить жюри, что это невозможно.

Но кроме этого Яну необходимо заполнить ведомость, поэтому ему придется воспользоваться услугами своей секретарши. За набор однозначного числа секретарша получает x рублей, за набор двузначного — y, а за набор ноля не получает ничего. Ян хочет минимизировать свои расходы, помогите ему.

Формат входного файла

В первой строке входного файла содержатся четыре целых числа: n — количество параметров, m — оценка которую хочет поставить жюри, x и y — стоимость набора однозначного и двузначного числа соответственно ($0 \le n, m, x, y \le 99$).

Следующие n строк содержат по два числа b_i и a_i — минимальная и максимальная оценка за i-ый элемент выступления ($0 \le b_i, a_i \le 99$).

Формат выходного файла

В выходной файл выведите одно число — минимально возможную суммарную стоимость выставления m баллов. Если баллы выставить нельзя, выведите в выходной файл число `-1`.

Примеры

figure.in	figure.out
2 12 1 99 1 10 1 10	2
2 7 1 99 1 10 0 10	1

Задача D. Лабиринт

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

Лабиринт — это сооружение, которое состоит из комнат и магических порталов. Каждый портал соединяет две комнаты. Портал выглядит как обычная дверь, но если эту дверь открыть, то мгновенно переносишься в ту комнату, в которую портал ведет.

На двери каждого портала со стороны комнаты написано некоторое слово. Истинное назначение этих пометок на дверях никому не известно. Некоторые исследователи считают, что их можно использовать для ориентирования в лабиринте. Другие думают, что смысл имеют не отдельные слова, написанные на дверях, а последовательности букв, которые получаются, если записать друг за другом несколько слов, которые написаны на дверях порталов, проходимых на пути из одной комнаты в другую. Такие последовательности букв в дальнейшем будем называть *метками путей*. Один из исследователей полагает, что важны метки не всех путей из одной комнаты в другую, а только наиболее короткие из них.

Ваша задача состоит в том, чтобы написать программу, которая по описанию лабиринта и номерам начальной и конечной комнат пути найдет самую короткую метку пути между этими комнатами. Если существует несколько самых коротких меток пути, то необходимо найти лексикографически наименьшую среди них.

Формат входного файла

Первая строка входного файла содержит два целых числа: n и m ($2 \le n \le 2000$, $1 \le m \le 50000$). Каждая из последующих m строк описывает один портал и содержит два числа: u и v ($1 \le u, v \le n$, $u \ne v$) — соответственно, номер комнаты, из которой портал выходит, и номер комнаты, в которую портал ведет, и слово w — пометку на двери портала, соединяющего эти комнаты (длина w находится в пределах от одного до десяти символов, w содержит только строчные буквы латинского алфавита).

Последняя строка входного файла содержит два целых числа: s и t ($1 \le s, t \le n$, $s \ne t$) — номера начальной и конечной комнаты пути, соответственно.

Формат выходного файла

Если из комнаты s нельзя добраться до комнаты t, то выведите в выходной файл слово `Impossible`. Иначе, выведите в выходной файл искомую метку пути.

Примеры

maze.in	maze.out
2 3 1 2 alpha 2 1 delta 1 2 gamma 1 2	alpha
2 2 1 2 alice 1 2 bob 1 2	bob
2 2 1 2 vice 1 2 versa 2 1	Impossible

Задача E. Принцип «горячей картошки»

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

Принцип *«горячей картошки»* представляет собой метод маршрутизации пакетов в сети. Его основными достоинствами являются отсутствие таблиц маршрутизации и отсутствие необходимости анализировать пакеты в процессе их маршрутизации.

Будем считать, что сеть состоит из n узлов, между некоторыми парами узлов установлены сетевые соединения. Каждый узел в сети имеет *расписание*. Расписание узла i представляет собой последовательность $a_{i,0}, a_{i,1}, \ldots, a_{i,l_i-1}$ номеров узлов. Сеть работает по тактам следующим образом.

В течение некоторого такта t в сети может находиться несколько пакетов. Каждый пакет имеет исходный узел и узел, в который он должен быть доставлен, в течение такта он находится в некотором узле. У каждого пакета есть время его появления, в начале этого такта он возникает в своем исходном узле. Если в течение такта t пакет находится в том узле, в который он должен быть доставлен, то считается, что он доставлен в момент t и он исчезает из сети. Иначе он переправляется в узел $a_{i,t \bmod l_i}$, где он оказывается в начале следующего такта. Если в начале некоторого такта несколько пакетов оказываются в одном узле, они все уничтожаются.

Главный недостаток этого принципа маршрутизации является следствием его основных преимуществ. Из-за отсутствия таблиц маршрутизации и анализа пакетов требуется очень аккуратно подбирать расписание вершин и время появления пакетов, чтобы все они были доставлены. Даже при удачном выборе часто нельзя доставить все пакеты.

Вам задано описание сети и набор пакетов. Для каждого пакета вы можете решить, будете ли вы пытаться его доставить. Вы должны выбрать максимальное количество пакетов, чтобы все они в процессе работы сети оказались доставлены.

Формат входного файла

Первая строка входного файла содержит число n — количество узлов в сети ($1 \le n \le 100$). Следующие n строк описывают узлы, i-й узел описывается сначала числом l_i — длиной своего расписания, а затем l_i числами: $a_{i,0}, a_{i,1}, \ldots, a_{i,l_i-1}$ ($1 \le l_i \le 8$, $a_{i,j} \ne i$).

Следующая строка входного файла содержит число p — количество пакетов ($1 \le p \le 100$), затем следует p строк, которые описывают пакеты. Каждый пакет описывается своим исходным узлом s_i, узлом, в который его следует доставить d_i и временем появления t_i ($s_i \ne d_i$, $0 \le t_i \le 1000$).

Формат выходного файла

На первой строке выходного файла выведите целое число k — максимальное количество пакетов, которое можно доставить. Вторая строка должна содержать k целых чисел — номера пакетов, которые можно доставить. Пакеты нумеруются от 1 до p в порядке, в котором они заданы во входном файле.

Примеры

hot.in	hot.out
4 2 2 3 3 1 1 3 3 4 4 2 1 3 3 1 3 0 3 1 2 4 2 3	2 2 3

Задача F. Машинное обучение

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

Машинное обучение — раздел теоретической информатики, который изучает возможности алгоритмов и компьютерных программ «*обучаться*». Обычно обучение происходит с использованием так называемых *обучающих примеров*. В этой задаче вам предстоит реализовать простейший вариант машинного обучения, натренировав детерминированный конечный автомат правильно распознавать слова из заданного множества.

Формально детерминированный конечный автомат представляет собой набор из четырех элементов: $\langle \Sigma, U, s, T, \varphi \rangle$, где Σ — конечное множество, которое представляет собой *входной алфавит* (в этой задаче будем полагать, что $\Sigma = \{0, 1\}$), U — это некоторое конечное множество *состояний*, $s \in U$ — *начальное состояние*, $T \subset U$ — множество *допускающих состояний*, а $\varphi : U \times \Sigma \to U$ представляет собой *функцию переходов*.

Входом для автомата является слово α, составленное из символов алфавита Σ. Исходно автомат находится в состоянии s. На каждом шаге он читает очередной символ c входного слова и изменяет свое состояние на $\varphi(u, c)$, где u — текущее состояние. После этого автомат переходит к следующему символу входного слова. Если когда слово целиком обработано автомат оказывается в допускающем состоянии, то говорят, что автомат *допускает* слово α, иначе говорят, что он его *не допускает*.

Разделим все слова длиной от 0 до заданного числа n на два множества: S^+ и S^-. Говорят, что автомат соответствует этому разбиению, если он допускает все слова из S^+ и не допускает все слова из S^-. Заметим, что слова длиннее n могут как допускаться, так и не допускаться автоматом.

Требуется построить автомат, соответствующий заданному разбиению, имеющий минимальное количество состояний.

Формат входного файла

Первая строка входного файла содержит число n ($1 \le n \le 12$). Следующие $2^{n+1} - 1$ строк описывают S^+ и S^-. Каждая строка содержит по одному слову, перед словом идет «`+`», если оно содержится в S^+, либо «`-`», если оно содержится в S^-. Слова упорядочены по длине, а при равной длине — лексикографически.

Формат выходного файла

На первой строке выходного файла выведите число u — минимальное количество состояний в автомате ($u \ge 1$) и s — номер начального состояния (состояния нумеруются от 1 до u).

Вторая строка должна содержать t — количество допускающих состояний, затем должно следовать t целых чисел — номера допускающих состояний.

Следующие u строк должны описывать переходы, каждая из этих строк должна содержать по два числа, i-я из строк должна содержать $\varphi(i, 0)$ и $\varphi(i, 1)$.

Примеры

learning.in	learning.out
2 + +0 -1 +00 -01 -10 -11	2 1 1 1 1 2 2 2