Problem A. Codecircles (Division 1 Only!)

Input file:	`codecircles.in`
Output file:	`codecircles.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

В популярной социальной сети «Codecircles» зарегистрированы n программистов, других пользователей в ней нет. Некоторые пары программистов не терпят друг друга, если a не может терпеть b, то и b не может терпеть a. Кругом в этой социальной сети называется такая циклическая последовательность не менее чем из трех различных программистов, что пара любых подряд идущих не терпит друг друга, а других таких пар среди программистов, входящих в круг, нет. Очевидно, что в любом круге из t программистов ровно t пар нетерпящих друг друга.

Известно, что все не настолько плохо в этой сети — **любые два круга имеют не более одного общего программиста**.

Чтобы как-то сдружить пользователей сети, администрация решила провести серию соревнований по программированию. Каждый день будет проведено ровно одно соревнование. Пользователи разбиваются на команды таким образом, что:

- каждый программист принадлежит ровно одной команде;
- нет пары нетерпящих друг друга программистов в одной команде;
- не должно быть пары соревнований, в которых принимают участие полностью одинаковые наборы команд.

Например, если $n = 2$ и эта пара программистов не является конфликтной, то можно провести всего два соревнования: в первом оба участника будут в одной команде, а во втором — будут в разных командах.

Помогите администрации выяснить, сколько соревнований можно провести, не нарушая предложенные правила.

Input

В первой строке записана пара целых чисел n, m ($1 \le n \le 500; 0 \le m \le \frac{n \cdot (n-1)}{2}$), n — количество программистов в сети «Codecircles», а m — количество нетерпящих друг друга пар. Далее следует m пар целых чисел, по одной паре в строке. Каждая пара (a_i, b_i) ($0 \le a_i, b_i \le n-1, a_i \ne b_i$) задает, что a_i-ый программист не терпит b_i-го (и наоборот). Программисты пронумерованы от 0 до $n-1$.

Входные данные не содержат одинаковых пар, даже если считать пары (a, b) и (b, a) одинаковыми.

Output

Выведите остаток количества соревнований при делении на 37493.

Examples

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

Problem B. Язык AZ

Input file:	az.in
Output file:	az.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Язык AZ — простой язык для записи бесполезных программ. Программа на языке AZ представляет собой строку, состоящую из одного или более последовательно записанных элементов. Есть два вида элементов: буква от "a" до "z" или корректная AZ-программа, которая взята в скобки, а перед левой открывающейся скобкой "(" находится целое число от 1 до 10^9.

Таким образом, грамматику языка можно записать как:

- <program>::=<element>|<program><element>
- <element>::='a'-'z'|<number>(<program>)
- <number>::= целое число от 1 до 10^9, записанное без лидирующих нулей.

Вот примеры корректных AZ-программ: «f», «ab2(b3(az))z», «1(a)».

При исполнении программы элементы исполняются один за другим слева направо. Элемент первого типа просто выводит на консоль соответствующую букву. Элемент второго типа исполняет программу в скобках столько раз, какое число написано перед левой открывающейся скобкой элемента. Данные выше примеры программ выведут «f», «abbazazazbazazazz» и «a» соответственно.

Интерпретатор AZ-2012 пока находится в стадии бета-тестирования и содержит известную ошибку реализации. Сразу после вывода k-ой буквы «a» на консоль он падает.

Напишите программу, которая по заданной AZ-программе и числу k выводит количества каждой из букв, которые появились на консоли при работе этой программы в интерпретаторе AZ-2012.

Input

Первая строка входных данных содержит корректную AZ-программу, длина которой не превосходит 10^5. Вторая строка содержит целое число k ($1 \le k \le 10^9$).

Output

Выведите t строк, где t — количество различных букв в выводе данной программы. Каждая строка должна иметь вид «c n», где c — это буква от "a" до "z", а n — количество ее вхождений в выводе программы. Информацию выводите в порядке возрастания номера буквы в алфавите. Если количество строго больше 10^{12}, то вместо количества выведите «-1» (без кавычек).

Examples

az.in	az.out
ab2(b3(az))z 3	a 3 b 2 z 1
ab2(b3(az))z 100	a 7 b 3 z 7

Problem C. Мины и ежи

Input file:	mines.in
Output file:	mines.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Неподалеку от некоторой деревни ее жители обнаружили страшные отголоски военных действий — минное поле. Чтобы как-то оградить местный скот от попадания на мины, было решено построить ограждение из колючей проволоки, окружающее все мины. Но как же это сделать? К счастью, в некоторых местах лежат тяжеленные противотанковые ежи (настолько тяжелые, что их никто не может сдвинуть со своих мест), за которые можно зацеплять колючую проволоку.

Так как у сельских жителей не очень много денег, было принято решение съездить в город и закупить как можно меньше колючей проволоки, необходимой для того, чтобы изолировать все мины от внешнего мира, последовательно привязывая проволоку к противотанковым ежам. Для того, чтобы конструкция получилась прочной и ее не унесло ветром, цепочка из проволоки должна начинаться и заканчиваться на одном и том же противотанковом еже. Шипы колючей проволоки довольно длинные, поэтому в случае, если проволока пройдет прямо над какой-то миной, на нее уже никто не наступит. Говоря более формально, проволоку нужно натянуть в виде замкнутой ломаной минимальной длины таким образом, чтобы все мины были ей изолированы. Мина считается изолированной, если любой уводящий бесконечно далеко путь от нее имеет общую точку с ломаной.

Ваша задача — помочь жителям деревни определить, сколько проволоки им нужно закупить.

Input

В первой строке даны два числа n и k ($3 \le n \le 300, 1 \le k \le 1000$) — количество ежей и мин соответственно. Далее в n строках по два целых числа x_i, y_i — координаты месторасположения ежей ($1 \le x_i, y_i \le 10^4$). В следующих k строках записано по два целых числа $\tilde{x}_i, \tilde{y}_i$ — координаты местоположения мин ($1 \le \tilde{x}_i, \tilde{y}_i \le 10^4$). Гарантируется, что все мины и ежи находятся в различных точках на плоскости.

Output

Вывести минимальную возможную длину колючей проволоки, которую потребуется купить, с точностью не менее 6 знаков после десятичной точки. Если окружить все мины невозможно, выведите одно число «-1» (без кавычек).

Examples

mines.in	mines.out
4 2 1 1 1 3 3 1 3 3 2 2 3 2	6.828427124746
3 1 1 1 1 3 1 5 1 4	4.000000000000

Note

Во втором примере ограждение вырождается в отрезок, проволоку нужно протянуть от второго ежа к третьему, и обратно.

Problem D. Самый длинный палиндром (Division 1 Only!)

Input file:	palindrome.in
Output file:	palindrome.out
Time limit:	3 seconds
Memory limit:	256 megabytes

Вам дан набор $S = (s_1, s_2, \ldots, s_n)$ из n строк. Все строки состоят из строчных латинских букв 'a'-'z'. Вы можете применять к набору две операции любое количество раз.

- Скопировать одну из строк из S и добавить копию в S (S может содержать одновременно любое количество одинаковых строк).
- Произвести конкатенацию любых двух строк из S и добавить результат в набор S.

Напомним, что палиндром — это строка, равная самой себе перевернутой.

Ваша цель — получить как можно более длинный палиндром, который входит как подстрока в одну из строк в S. Найдите длину этого палиндрома.

Заметим, что возможны случаи, когда ответ неопределен, потому что возможно получить палиндром с длиной, большей любого наперед заданного числа. В таких случаях условимся считать, что длина самого длинного палиндрома равна бесконечности.

Input

Первая строка содержит целое число n ($1 \le n \le 100$) — изначальное количество строк в S. Следующие n строк содержат строки s_i, $1 \le i \le n$, состоящие из строчных латинских букв 'a'-'z'. Каждая строка имеет длину от 1 до 1000 включительно. Могут быть одинаковые строки.

Output

Выведите длину самого длинного палиндрома, который может быть получен описанным образом из заданного набора S. Если эта длина равна бесконечности, выведите "-1".

Examples

palindrome.in	palindrome.out
3 abc abacde ecab	7
1 ab	-1

Note

В первом примере объединяем "ecab" и "abacde", получаем "e**cababac**de" с палиндромом "cababac" длины 7.

Во втором примере можно сделать любое количество копий строки "ab" и получать строки вида "ab", "abab", "ababab", и т.д. Эти строки не являются палиндромами, но содержат палиндромы, например, "a", "aba", "ababa", и т.д. Таким образом, возможно получить палиндром любой нечетной длины и ответ равен бесконечности.

Problem E. Переливания

Input file:	pour.in
Output file:	pour.out
Time limit:	2 seconds
Memory limit:	256 megabytes

На столе стоят n одинаковых стаканов. В каждый из них налито a миллилитров жидкости. Всего в каждый стакан помещается b миллилитров жидкости. Двое играют в следующую игру. Игроки ходят по очереди. За ход необходимо взять любой стакан и перелить **всю** жидкость из него в любой другой стакан, который при этом не переполнится. Пустой стакан после хода выбрасывается и больше не участвует в игре. Проигрывает тот, кто не сможет сделать ход. Определите, кто выиграет при правильной игре.

Input

В первой строке заданы три целых числа n, a и b ($1 \le n, a, b \le 1000$, $a \le b$) — количество стаканов, изначальное количество жидкости в каждом из стаканов и максимальное количество жидкости, которое помещается в стакан.

Output

Выведите номер игрока (1 или 2), который выиграет оптимальной игре обоих.

Examples

pour.in	pour.out
5 1 3	1
2 3 5	2

Problem F. Сапсан (Division 1 Only!)

Input file:	`sapsan.in`
Output file:	`sapsan.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

В Берляндии в рамках программы по совершенствованию железнодорожного сообщения скоро появится новый скоростной поезд, который будет называться «Сапсан». Запуск этого поезда значительно упростит перемещение между двумя самыми главными городами страны. Однако для запуска скоростного поезда не предполагается вводить в строй новые дороги, а использовать уже имеющиеся, что кроет в себе некоторые проблемы.

Задана карта железных дорог, соединяющих n станций. В стране есть $n-1$ двунаправленная железная дорога. Сеть дорог устроена таким образом, что существует способ добраться по ним между любой парой станций. Каждая дорога имеет некоторую длину l_i метров и представляет из себя два пути в противоположных направлениях (т.е. два поезда могут двигаться одновременно в разных направлениях по этой дороге). Имеется m обыкновенных поездов, про каждый из которых известны станция отправления s_i, станция назначения d_i и время отправления t_i (в секундах). Поезда ездят каждый день, при этом продолжительность дня в Берляндии можно считать очень большой, т.е. все поезда успевают доехать до станции назначения до конца дня. Поезда не тратят время на остановки на станциях. Пассажиры спрыгивают и запрыгивают на ходу, благо скорости всех поездов одинаковые и равны одному метру в секунду. Поезда можно считать материальными точками. При таких условиях легко понять, что поезда не могут помешать движению друг друга.

Однако, после запуска нового скоростного поезда ситуация может осложниться, потому что новый поезд двигается со скоростью $v > 1$ метров в секунду (недаром он скоростной). Теперь некоторые существующие поезда могут помешать движению Сапсана. Поезд мешает Сапсану, если:

- Сапсан догоняет поезд в некотором месте на дороге (т.е. находится с ним в одной точке, отличной от станции, и движется с ним в одном направлении)
- Сапсан догоняет поезд на станции, и при этом в эту станцию Сапсан и поезд приехали по одной дороге и выедут по одной дороге.

Для поезда Сапсан известны станция отправления a и станция назначения b, а также интервал времен $[t_l, t_r]$, в котором может быть выбрано время его отправления. Время отправления Сапсана должно быть целым числом. Если поезд мешает Сапсану, то поезд должен подождать время, нужное для того, чтобы пропустить Сапсан. Поезд должен ждать в **первой** на своем пути станции, через которую проезжает Сапсан. На одной станции может ждать неограниченное количество поездов, и это никак не мешает движению через эту станцию других поездов.

Необходимо выбрать время для отправления Сапсана так, чтобы минимизировать суммарное время ожидания всех поездов.

Input

В первой строке записано целое число n ($2 \le n \le 10^5$) — количество станций. В каждой из следующих $n-1$ строк записано по три целых числа x_i, y_i, l_i ($1 \le x_i, y_i \le n, 1 \le l_i \le 10^4$) — номера станций, соединенных дорогой, и длина дороги. В следующей строке записано целое число m — количество поездов ($0 \le m \le 10^5$). В следующих m строках записано по три целых числа s_i, d_i, t_i ($1 \le s_i, d_i \le n, s_i \ne d_i, 0 \le t_i \le 10^9$). В последней строке описывается поезд Сапсан пятью целыми числами a, b, v, t_l, t_r ($1 \le a, b \le n, a \ne b, 1 < v \le 100, 0 \le t_l \le t_r \le 10^9$).

Output

Выведите оптимальное время отправления Сапсана. Если оптимальных решений несколько, выведите любое из них.

Examples

sapsan.in	sapsan.out
6 1 2 3 2 3 6 3 4 4 2 5 1 3 6 1 1 5 6 0 1 4 2 0 2	0
6 1 2 3 2 3 6 3 4 4 2 5 1 3 6 1 1 5 6 0 1 4 2 1 3	3

Problem G. Второй дивизион (Division 1 Only!)

Input file:	`division2.in`
Output file:	`division2.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Во втором дивизионе Берляндской Футбольной Лиги проходит необычный чемпионат по футболу. Его необычность заключается в том, что каждая команда может набирать очки только в домашних играх — 3 за победу, 1 за ничью и 0 за поражение. В гостевых матчах команда лишь может помешать набрать очки сопернику, который играет дома.

Дана турнирная таблица чемпионата, состоящая из двух столбцов — «название команды» и «количество набранных очков». В зависимости от занятого места команда получает больше или меньше призовых. Но есть условие, что команде желательно не попадать в k лучших, иначе она выйдет в первый дивизион, на участие в котором у нее нет денег и это, скорее всего, приведет к банкротству клуба. Будем считать, что команды пронумерованы числами от 1 до n в том порядке, в котором они перечислены в таблице.

Известны вероятности исходов матчей для всех команд. Для i-й команды про каждого соперника с номером j дана вероятность победы над ним дома — $pwin_{ij}$, и вероятность ничьей — $pdraw_{ij}$ ($0 \le pwin_{ij} + pdraw_{ij} \le 100$, вероятности даны в процентах).

Известно, что с одним соперником нельзя сыграть дома более одного раза. Для определенности дана таблица $played_{ij}$ — сколько раз команда i уже сыграла с командой j у себя на поле (число 0 или 1).

Командам не обязательно играть все домашние игры, за каждый не сыгранный матч будет зачтено техническое поражение (при этом команда, с которой матч не был сыгран, ничего не приобретает и не теряет — в силу описанных выше необычных правил чемпионата). При этом никакая команда не может отказаться от гостевой игры.

При подведении итогов чемпионата команды располагаются в турнирной таблице по убыванию количества набранных очков. В случае равенства очков в ход идут дополнительные показатели (разница забитых и пропущенных мячей и т.п.).

Вы являетесь менеджером одной из команд в чемпионате. Вы должны выбрать, с какими командами еще стоит сыграть, чтобы занять место как можно выше, но при этом ниже k-го (таким образом, приоритетной целью является $(k+1)$-е место). Можно считать, что все остальные соперники не пользуются никакой стратегией выбора команд, с которыми еще будут играть дома, то есть для них вероятность сыграть с любым из оставшихся соперников равна 50%. Можно считать, что ваша команда крайне плохо играет в атаке, поэтому при равенстве набранных очков с другими командами она окажется ниже в турнирной таблице из-за худших дополнительных показателей.

Таким образом, нужно доиграть некоторые матчи, чтобы занять $(k+1)$-е место с максимально возможной вероятностью. Какая при этом будет вероятность выйти в первый дивизион, значения не имеет.

Input

В первой строке два числа n и k — число команд в чемпионате и число команд, выходящих в первый дивизион по итогам турнира, соответственно ($2 \le n \le 18, 1 \le k \le n-1$), далее следуют n строк вида “name points”, где “name” — это название команды, а “points” — количество набранных ею очков. В следующих n строках описана таблица $pwin$ — по n целых чисел в каждой строке. Далее в n строках описана таблица $pdraw$ — по n целых чисел в каждой строке. Следующие n строк по n целых чисел описывают таблицу $played$ ($0 \le pwin_{ij} \le 100, 0 \le pdraw_{ij} \le 100, 0 \le played_{ij} \le 1$). Данные о количестве очков у каждой команды и информация о уже сыгранных матчах не противоречат

друг другу, т.е. эти матчи могли закончится таким образом, чтобы у каждой команды оказалось указанное количество очков. В последней строке задано название вашей команды, оно совпадает с одним из названий в турнирной таблице. Все названия команд различны и представляют собой непустые строки из строчных и заглавных букв латинского алфавита длины не более 100.

Output

В первой строке выходного файла необходимо вывести единственное число m — количество команд, с которыми вашей команде еще следует сыграть. В следующих m строках выведите названия этих команд, по одному в строке. Если ваша команда уже сыграла с i-ой, то i-ую команду выводить не нужно. В случае, когда оптимальных решений несколько, выведите любое из них.

Examples

division2.in	division2.out
4 1 Rotor 9 Fakel 6 Sokol 3 Gazmyas 0 0 50 50 100 50 0 50 100 50 50 0 50 0 0 50 0 0 50 50 0 50 0 50 0 50 50 0 50 0 0 50 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 Sokol	2 Fakel Gazmyas

Note

В примере вашей команде нужно обязательно сыграть оба оставшихся матча, в противном случае можно набрать не более 6 очков, и занять второе место будет невозможно независимо от игр соперников.

Problem H. Фонари

Input file:	`lamps.in`
Output file:	`lamps.out`
Time limit:	6 seconds
Memory limit:	256 megabytes

На очень длинной прямолинейной автомобильной дороге из Бирляндии в Берляндию имеется участок, освещенный n фонарями. Если считать, что вдоль дороги введена ось координат, то получается, что фонарь с номером i освещает дорогу в отрезке $[a_i, b_i]$. Фонари расположены так, что при движении из Бирляндии (которой на оси соответствует $-\infty$) в Берляндию (которой соответствует $+\infty$) сначала идет неосвещенная дорога, потом следует освещенный участок дороги без каких-либо разрывов (т.е. каждое место на нем освещается хотя бы одним из фонарей), а затем неосвещенная дорога до самой Берляндии.

Специальные службы Берляндии получили информацию о том, что следующей ночью Бирляндский шпион попытается приехать в их страну по этой дороге. Чтобы не быть замеченным, шпион будет использовать специальное средство — зелье невидимости. Алгоритм использования зелья прописан в секретной инструкции и выглядит следующим образом:

- Когда при своем движении шпион переходит из неосвещенной зоны в освещенную, он выпивает зелье и становится невидимым.
- Когда невидимый шпион переходит из освещенной зоны в неосвещенную, он должен незамедлительно выпить зелье с обратным эффектом и стать видимым. Этот пункт инструкции крайне важен, потому что длительное нахождение в состоянии невидимости может угрожать жизни шпиона.

Для того, чтобы не дать шпиону пробраться в Берляндию незамеченным, агент службы безопасности должен выключить некоторые фонари. На выключение фонаря с номером i уходит t_i минут. Известно, что у Бирляндского шпиона имеется **два** комплекта зелий: один основной, другой запасной. Поэтому задача агента состоит в том, чтобы выключить такой набор фонарей, что при движении по дороге из Бирляндии в Берляндию вражескому шпиону потребуется не менее **трех** комплектов зелий. Естественно, что среди всех таких наборов фонарей агент должен выбрать такой, что суммарное время выключения всех фонарей в нем как можно меньше.

Input

Первая строка содержит цело число n ($1 \le n \le 100000$) — количество фонарей. Далее n строк содержат описания фонарей по одному в строке в виде трех целых чисел a_i, b_i, t_i ($-10^9 \le a_i \le b_i \le 10^9, 1 \le t_i \le 10^9$). Отрезки могут произвольным образом пересекаться и даже совпадать.

Output

Выведите в первую строку наименьшее суммарное время выключения нужных фонарей. Вторая строка должна содержать количество фонарей в искомом наборе. В третью строку выведите последовательность номеров фонарей в искомом наборе. Фонари нумеруются с 1 в порядке их появления во входных данных. Номера фонарей выводите в любом порядке. Если ответов несколько, выведите любой. Если решения не существует, то выведите единственное число «`-1`» (без кавычек).

Examples

lamps.in	lamps.out
8 6 7 2 7 8 8 2 6 9 0 1 7 1 6 5 1 3 4 5 6 8 1 5 2	13 4 1 5 6 8

Problem I. Обувной вопрос

Input file:	`shoes.in`
Output file:	`shoes.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Король Берляндии Берл XIX решил направиться с визитом в соседнюю страну Бирляндию. Отношения между странами нуждаются в укреплении, поэтому Берл заинтересован в том, чтобы его визит длился как можно дольше. Бирляндия состоит из n городов, соединенных $n-1$ двусторонней дорогой таким образом, что существует способ добраться по этим дорогам из каждого города до любого другого.

Король может начать свой путь в любом городе. Далее он может каждый раз переезжать из текущего города в некоторый соединенный с ним дорогой город. Берл не может посещать один и тот же город более одного раза. То, в каком городе закончится маршрут короля, значения не имеет. Гораздо более важно то, что политическая обстановка в Бирляндии весьма напряженная. Один важный вопрос разделил страну на два противоборствующих лагеря. Люди не могут договориться, какую ногу следует обувать первой: правую или левую? Приверженцы правой ноги заявляют, что в этом случае человеку будет сопутствовать удача. Приверженцы левой ноги говорят, что в этом случае человек будет здоров, да и неудобно потом прыгать на левой необутой ноге за забытыми вещами. Про каждый город известно, какую позицию занимают его жители.

Берл не хочет выказывать предпочтение какой-либо из группировок, поэтому в его пути количество городов, выступающих за левую ногу, должно быть равно количеству городов, выступающих за правую. Помогите королю найти как можно более длинный такой маршрут.

Input

В первой строке записано целое число n ($1 \le n \le 10^5$). Во второй строке дана строка из n символов '`L`' или '`R`'. Если i-й символ этой строки равен '`L`', то город с номером i выступает за левую ногу, иначе — за правую. В следующих $n-1$ строках записано по два целых числа u_i, v_i ($1 \le u_i, v_i \le n$) — номера городов, соединенных i-й дорогой.

Output

В первой строке выведите наибольшую возможную длину маршрута короля (если нет ни одного подходящего маршрута, выводите 0). Под длиной маршрута подразумевается количество городов, через которые проходит маршрут. Во второй строке выведите оптимальный маршрут, перечисляя номера составляющих его городов в порядке прохождения. Если оптимальных маршрутов несколько, выведите любой.

Examples

shoes.in	shoes.out
3 LRR 1 2 1 3	2 1 2
5 LRLLR 1 2 2 3 3 4 2 5	4 5 2 3 4

Problem J. Уничтожить жуков

Input file:	`beetles.in`
Output file:	`beetles.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

У преуспевающего фермера Анатолия есть несколько картофельных полей, приносящих ему стабильный и весьма неплохой доход. Однако ничего в этом мире не дается даром, в том числе и хороший урожай картофеля. Особенно тяжело защищать поля от колорадского жука. Раньше Анатолий привлекал для уничтожения жуков гастарбайтеров из соседней страны, но теперь их депортировали на родину. Поэтому Анатолий поехал в город и приобрел там новейшее чудо техники — робота, уничтожающего колорадского жука. Теперь все поля должны быть обработаны этим роботом.

Всего у Анатолия n картофельных полей, некоторые из которых соединены дорогами. По каждой дороге робот может передвигаться в обе стороны, причем на проезд по i-й из них он тратит d_i часов. Сеть дорог устроена таким образом, что существует ровно один способ добраться от любого поля до любого другого.

Робот должен обойти все поля, побывав на каждом хотя бы один раз. Каждое поле должно быть обработано роботом в одно из посещений этого поля. Обработка i-го поля занимает непрерывный отрезок времени длиной t_i часов. Робот начинает свой обход в момент времени 0 с поля номер 1 и должен вернуться на него после обработки всех полей. Кроме того, есть дополнительное ограничение: по каждой дороге робот может пройти только один раз в каждом из направлений.

Анатолию осталось только записать в память робота информацию о том, в каком порядке обходить и обрабатывать поля. Положение осложняется тем, что все поля неравнозначны: какие-то больше по размеру, какие-то больше поражены жуком и т.п. Поэтому разные программы действий для робота будут иметь разную эффективность. Анатолий оценил, что если обработка поля с номером i будет закончена в момент времени t, то его потери на этом поле составят $k_i \cdot t$ рублей. Помогите Анатолию задать роботу такую программу, чтобы минимизировать суммарные потери.

Input

В первой строке записано целое число n ($2 \le n \le 10^5$). Во второй строке записано n целых чисел t_i ($1 \le t_i \le 10^4$). В третьей строке записано n целых чисел k_i ($1 \le k_i \le 10^4$). Следующие $n-1$ строка описывают дороги: в i-й из этих строк содержится три целых числа: номера полей u_i и v_i, соединенных дорогой, и время проезда робота по дороге d_i ($1 \le u_i, v_i \le n, 1 \le d_i \le 10^4$).

Output

В первой строке выведите минимальную возможную сумму потерь. Далее выведите $3n-2$ строки, описывающие действия робота. Для команды перемещения в i-е поле «M i», для команды обработки i-го поля выводите «P i». Если оптимальных решений несколько, выведите любое из них.

Examples

beetles.in	beetles.out
3 2 3 4 1 1 1 1 2 1 1 3 1	20 P 1 M 2 P 2 M 1 M 3 P 3 M 1
3 2 3 4 1 10 1 1 2 1 1 3 1	59 M 2 P 2 M 1 P 1 M 3 P 3 M 1

Problem K. Телевышка

Input file:	tower.in
Output file:	tower.out
Time limit:	7 seconds
Memory limit:	256 megabytes

У жителей Флатландии большая радость. Скоро в домах всех ее обитателей появится телевидение. Для того, чтобы это произошло, осталось сделать немного: правильно настроить работу единственной телевышки, которая расположена в точке с координатами $(0, 0)$. В силу конструктивных особенностей телевышки, зона ее вещания будет представлять собой правильный k-угольник с центром в $(0, 0)$. При этом размер и поворот этого k-угольника относительно этой точки могут быть выбраны любыми. Естественно, зона вещания должна охватывать все имеющиеся во Флатландии города. В стране n городов, i-й из которых располагается в точке с координатами (x_i, y_i). При выбранном повороте и размере k-угольника зоны вещания всего города должны находиться внутри него или на его границе. Чем больше размер зоны вещания, тем больше энергии тратится на работу телевышки, поэтому власти страны хотят найти наименьший радиус зоны вещания (радиус окружности, описанной около правильного k-угольника), при котором можно найти такой ее поворот, что телевидение будет доступно во всех городах.

Input

В первой строке записана пара целых чисел n и k ($1 \le n \le 50000, 3 \le k \le 50000$). В каждой из следующих n строк содержится по два целых числа x_i, y_i ($-10^6 \le x_i, y_i \le 10^6$) — координаты городов. Никакие два города не находятся в одной точке. В точке $(0, 0)$ города нет.

Output

Выведите ответ на задачу с точностью не менее чем 10^{-9} (т.е. абсолютная или относительная погрешность не должны превосходить этой величины).

Examples

tower.in	tower.out
1 4 1 1	1.414213562373095
3 4 1 2 2 1 -3 0	3.0

Problem L. Economical Crisis (Division 2 Only!)

Input file:	`crisis.in`
Output file:	`crisis.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Джон прочитал в газете о надвигающемся кризисе и решил, что пришла пора массово закупать продукты в ближайшем супермаркете, пока они не подорожали. У Джона есть карта скидок из этого супермаркета, так что некоторые товары он может покупать со скидкой.

По заданному списку покупок, которые планирует совершить Джон, а также списку товаров в супермаркете определите сумму, которую сэкономит Джон благодаря карте скидок.

Input

В первой строке входного файла задано целое число K ($1 \le K \le 10$) — количество тестовых примеров.

Далее следуют сами тестовые примеры. Первая строка каждого тестового примера содержит два целых числа N и M ($1 \le N, M \le 100$) — количество различных продуктов в супермаркете и список продуктов, которые Джон собирается купить.

Далее следуют N строк, каждая из которых описывает наличие продукта в магазине. Каждая из этих строк содержит целое число n_i ($n_i < 1000$) — количество упаковок этого продукта в наличии, два вещественных числа, заданных с двумя знаками после десятичной точки — ценами p_i и c_i в долларах для обычного покупателя и для владельца карты скидок ($0.00 \le p_i, c_i \le 99.99,\ c_i \le p_i$). Далее идёт наименование товара s_i, которое может состоять из не менее, чем одной, и не более ста строчных и прописных латинских букв, а также содержать пробелы (не в начале и не в конце строки).

Далее следуют M строк — список покупок, которые планирует совершить Джон. Каждая строка содержит целое число m_j ($m_i < 1000$) — количество упаковок продукта, планируемое Джоном к покупке и название соответствующего продукта s_j, состоящее из строчных и прописных латинских букв, а также, возможно, содержащее пробелы (не в начале и не в конце). Название содержит не менее одного и не более 100 символов.

Если названия продукта в магазине и в списке покупок совпадают (без учёта капитализации), считается, что этот продукт в магазине есть.

Output

Для каждого тестового примера в отдельной строке выведите одно число — точную сумму, которую Джон сэкономил, используя карту скидок. Число должно содержать ровно два знака после десятичной точки.

Example

crisis.in	crisis.out
1 2 3 3 3.00 3.00 Coca Cola 2 6.00 4.50 meat of kangaroo 1 coca cola 3 meat of kangaroo 1 russian vodka	3.00

Problem M. DNA sequences (Division 2 Only!)

Input file:	`dna.in`
Output file:	`dna.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Система для автоматического анализа ДНК работает следующим образом. В результате обработки статистической информации система выделяет цепочки аминокислот, которые предположительно образуют ген. После чего учёные проверяют выданные цепочки на коректность. При этом система выводит последовательность аминокислот в виде строки, в которой предполагаемые цепочки генов отмечены дугой, соединяющей их начало и конец.

При этом возможна ситуация, когда некоторые цепочки перекрывают друг друга (учитывая то, что выдаваемая информация — гипотезы, это вполне возможно). В такой ситуации, чтобы избежать пересечения дуг, система строит дуги сверху и снизу. Но иногда даже этого бывает недостаточно.

Ваша задача — по заданным начальным и конечным позициям выданный системой цепочек выяснить, возможно ли выделить все цепочки так, чтобы никакие две дуги не пересекались.

Input

Входной файл состоит из не более, чем 120 тестовых примеров. Каждый тестовый пример расположен в одной строке и начинается с целого числа k ($0 < k < 10000$) — количество предполагаемых цепочек в последовательности аминокислот. Далее следуют k пар целых чисел — номера a_i и b_i позиций, являющихся концами i-го предполагаемого гена ($0 \le a_i, b_i \le 2k - 1$). Гарантируется, что все a_i и b_i попарно различны (то есть никакие две цепочки не имеют общих концов). Входной файл заканчивается тестовым примером с $k = 0$, обрабатывать который не требуется.

Output

Для каждого тестового примера в отдельной строке выведите "`possible`", если возможно выделить предполагаемые гены дугами так, чтобы они не пересекались, и "`impossible`" в противном случае.

Example

dna.in	dna.out
3 0 4 1 3 2 5 3 0 3 1 4 2 5 0	possible impossible

Problem N. Extrusion (Division 2 Only!)

Input file:	`extrusion.in`
Output file:	`extrusion.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Металлургическая компания «Рвалл и Металл» специализируется на изготовлении стальных балок с заданным профилем. Технология изготовления может быть описана следующим образом: в тонкой пластине из огнеупорного материала делается отверстие, край которого является многоугольником, после чего заготовка из полурасплавленного металла протягивается через это отверстие. Получается балка, профиль которой соответствует заданному отверстию.

По объёму имеющегося металла и параметрам отверстия вычислите длину получившейся балки.

Input

В первой строке входного файла задано одно целое число T ($1 \le T \le 10$) — количество тестовых примеров.

Первая строка каждого тестового примера содержит одно целое число N ($3 \le N \le 20$) — количество вершин многоугольника. В последующих N строках заданы по два вещественных числа (x_i, y_i) — координаты вершин многоугольника, заданные в порядке обхода по часовой стрелке ($0 \le x_i, y_i \le 10.0$). Гарантируется, что площадь многоугольника не равна нулю. Завершается тестовый пример строкой, содержащей вещественное число v_i — объём имеющегося в наличии металла $0 < v_i \le 100.0$.

Output

Для каждого тестового примера в отдельной строке выведите с точностью 10^{-2} одно вещественное число — длину получившейся балки.

Example

extrusion.in	extrusion.out
2 4 0.0 0.0 0.0 0.1 0.1 0.1 0.1 0.0 1.0 7 0.5 1.25 0.9 1.6 0.9 1.1 0.85 1.0 0.9 0.85 0.9 0.5 0.5 0.75 100.0	100.000 318.73

Problem O. Yet Another Game (Division 2 Only!)

Input file:	`game.in`
Output file:	`game.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Рассмотрим игру для двух игроков. Изначально задано некоторое количество кучек камней. Игра состоит из двух этапов. На первом этапе первый игрок может убрать какое-то количество кучек (но не все), или не убирать ни одной кучки, после чего второй игрок также может убрать какое-то количество кучек (но не все) или не убирать ни одной кучки.

На втором этапе игроки делают ходы по очереди. Каждый игрок во время своего хода может выбрать некую кучку камней и взять любое ненулевое количество камней из этой кучки. Игрок, который забрал последний камень, объявляется победителем.

По заданной начальной позиции в игре вычислите, может ли первый игрок обеспечить себе победу, и, если может, какое наименьшее число камней он должен убрать во время первого этапа для этого.

Input

Первая строка входного файла содержит одно целое число T — количество тестовых примеров ($1 \le T \le 30$). Каждый тестовый пример состоит из двух строк. Первая содержит целое число N ($1 \le n \le 100$) — изначальное количество кучек камней.. В следующей строке заданы N целых чисел p_i ($1 \le p_i \le 1000$), i-е из которых задаёт количество камней в i-й кучке.

Output

Для каждого тестового примера выведите в отдельной строке одно целое число — минимальное количество камней, которое может убрать с доски первый игрок во время первой стадии игры для того, чтобы обеспечить себе победу, или -1 в случае, если не существует такого хода на первой стадии игры, который бы обеспечил победу первого игрока.

Example

game.in	game.out
1 3 2 1 3	1