Задача A. Вредные привычки

Имя входного файла:	`addict.in`
Имя выходного файла:	`addict.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Почти все люди имеют различные вредные привычки. Некоторые из них довольно безвредны (или, по крайней мере, считаются такими), а другие могут представлять реальную опасность для здоровья или даже для жизни людей.

Однако расстаться с вредными привычками непросто, причем часто чем опаснее привычка, тем сложнее от нее отказаться. Иногда для борьбы с так называемым *синдромом отмены* люди даже специально приобретают некоторые новые вредные привычки, чтобы хоть как-то скомпенсировать недостающие ощущения.

Ваня планирует избавиться от большинства своих привычек. Он классифицировал их и для каждой вредной привычки знает ее силу s_i.

Он решил избавляться от одной привычки в неделю. Но поскольку избавиться от привычки тяжело, то иногда ему придется одновременно восстановить некоторые из своих старых, уже казалось бы забытых, привычек. Ваня провел испытания своей силы воли и понял, что он сможет двигаться к своей цели, если после отказа от какой-то вредной привычки и восстановления некоторых других, суммарная сила его вредных привычек уменьшится не более чем на d. При этом Ваня никогда не будет делать так, чтобы суммарная сила его вредных привычек увеличилась.

Помогите Ване избавиться от максимального количества вредных привычек, и сделать это за минимальное количество недель.

Формат входного файла

Первая строка входного файла содержит число n — количество Ваниных вредных привычек ($1 \le n \le 1000$) и d ($1 \le d \le 10000$). Следующие n строк описывают привычки — каждая строка содержит название привычки (строку из не более чем 40 символов английского алфавита) и ее силу (положительное целое число, не превосходящее 10000).

Формат выходного файла

На первой строке выходного файла выведите два числа: k — максимальное количество привычек, от которых Ване удастся избавиться, и t — минимальное количество недель, которое ему для этого потребуется. Следующие k строк должны содержать привычки, от которых удастся избавиться, по одной на строке, отсортированные в алфавитном порядке.

Примеры

addict.in	addict.out
5 3 Fighting 7 Coffee 1 TV 4 Oversleep 5 Contests 25	4 9 Coffee Fighting Oversleep TV

Задача B. Обобщенные числа Фибоначчи

Имя входного файла:	`fib.in`
Имя выходного файла:	`fib.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Обобщенными числами Фибоначчи $F_n^{(k)}$ называют следующую последовательность: $F_1^{(k)} = F_2^{(k)} = \ldots = F_k^{(k)} = 1$, $F_i^{(k)} = \sum_{j=1}^{k} F_{i-j}^{(k)}$, $i > k$.

Ваша задача — вычислить остаток от деления $F_n^{(k)}$ на p.

Формат входного файла

Входной файл содержит три целых числа: n, k и p ($1 \le n, k \le 10^6$, $2 \le p \le 10^9$).

Формат выходного файла

В выходной файл выведите $F_n^{(k)} \bmod p$.

Примеры

fib.in	fib.out
3 2 10	2
10 2 10	5

Задача C. Игра средней оплаты

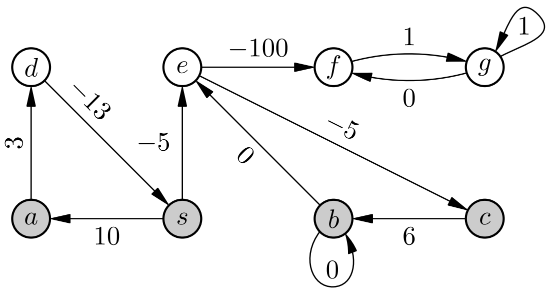
Имя входного файла:	`mean.in`
Имя выходного файла:	`mean.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	64 мегабайта

Игра средней оплаты — это игра для двух игроков, которая ведется на ориентированном графе. Вершины графа разделены на два множества: A и B. Первому игроку принадлежат вершины из множества A, а второму игроку — вершины из множества B. Каждая дуга uv граф помечена некоторым целым числом w_{uv}.

Игра протекает следующим образом. Исходно специальная фишка помещается в выделенную вершину s графа. Затем игроки делают ходы, перемещая фишку по графу. Если фишка находится в вершине из множества A, то фишку перемещает первый игрок, а если в вершине из множества B — то второй игрок. Игроки поддерживают специальный счетчик z, в котором ведется суммирование чисел, записанных на дугах, по которым перемещается фишка. Исходно $z = 0$, после того как фишка перемещается по дуге uv, выполняется присваивание $z \leftarrow z + w_{uv}$. Игра продолжается до бесконечности.

Цель первого игрока — добиться, чтобы $z \to +\infty$ (иначе говоря, добиться, чтобы для любого числа x с некоторого момента игры z никогда не становился меньше чем x). Цель второго игрока — добиться, чтобы $z \to -\infty$. Если ни один из игроков не может добиться успеха, игра считается ничейной.

Например, рассмотрим игру, граф которой изображен на рисунке. Закрашенные вершины принадлежат первом игроку, а незакрашенные — второму. Исходно фишка располагается в вершине s. Эта игра выигрышная для первого игрока. Он должен переместить фишку в вершину e и, второй игрок может либо перейти в положительный цикл $f \to g \to f$, из которого нет выхода, или переместить фишку в вершину c, из которой первый игрок переместит ее в b, и затем опять в e, добавляя $+1$ к z.



Можно показать, что если игра является выигрышной для одного из игроков, то у него существует *детерминированная стратегия*. Такая стратегия заключается в том, что для каждой вершины из тех, которые принадлежат игроку, указывается дуга, по которой следует перемещать фишку, если она оказалась в этой вершине. Стратегия не зависит ни от текущего значения z, ни от другой информации о предыдущем ходе игры.

Петя собирается поиграть в эту игру с Васей. Петя будет первым игроком, а Вася — вторым. Игроки выбрали граф для игры и стартовую вершину. Теперь Петя считает, что он выиграет и игре и выбрал себе выигрышную стратегию. Но он не до конца уверен. Он просит вас проверить, действительно ли выбранная им стратегия является выигрышной, то есть гарантирует, что $z \to +\infty$ вне зависимости от действий Васи.

Формат входного файла

Первая строка входного файла содержит два целых числа n и m — количество вершин и ребер в графе, соответственно. ($1 \le n \le 2\,000$, $1 \le m \le 10\,000$). вторая строка содержит n латинских букв, которые описывают владельцев вершин, вершины, принадлежащие Пете, помечены как '`A`', а вершины, принадлежащие Васе — как '`B`'. Третья строка содержит $0s$ — номер стартовой вершины.

Следующие m строк описывают дуги графа. Каждая дуга задается вершиной, в которой она начинается, вершиной, в которую она ведет, и числом, которым она помечена. Пометки дуг не превосходят 10^5 по абсолютной величине. Из каждой вершины исходит хотя бы одна дуга.

Последняя строка описывает стратегию Пети. Они содержит n целых чисел. Для каждой вершины задано либо число 0, если соответствующая вершина принадлежит Васе, либо номер дуги, по которой Петя собирается пойти, если фишка окажется в этой вершине. Дуги нумеруются с 1 в том порядке, в котором они заданы во входном файле.

Формат выходного файла

Если Петина стратегия действительно является выигрышной, выведите "`Win`" на первой строке выходного файла. В противном случае выведите либо "`Lose`", если Вася может выиграть, если Петя будет пользоваться этой стратегией, либо "`Draw`", если Вася выиграть не может, но может свести игру к ничьей. В этих случаях на второй строке выведите Васину выигрышную либо ничейную стратегию (против заданной Петиной) в том же формате, в котором во входном файле задана Петина стратегия.

Пример

mean.in	mean.out
8 12 AAAABBBB 2 2 1 10 3 3 0 4 3 6 1 5 3 5 2 -13 2 6 -5 3 6 0 6 4 -5 6 7 -100 7 8 1 8 7 0 8 8 1 4 6 7 3 0 0 0 0	Win
8 12 AAAABBBB 2 2 1 10 3 3 0 4 3 6 1 5 3 5 2 -13 2 6 -5 3 6 0 6 4 -5 6 7 -100 7 8 1 8 7 0 8 8 1 4 1 7 3 0 0 0 0	Draw 0 0 0 0 5 8 10 11

Задача D. Преобразователь строк

Имя входного файла:	`morpher.in`
Имя выходного файла:	`morpher.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Преобразователь строк — специальная программа, получающая на вход строку и выдающая на выход также строку. В процессе обработки к некоторым подстрокам исходной строки применяются операции двух типов:

- разворот (обозначается символом `R`) — из строки $s_1 s_2 \ldots s_{n-1} s_n$ получается строка $s_n s_{n-1} \ldots s_2 s_1$;
- сортировка (обозначается символом `S`) — из строки $s_1 s_2 \ldots s_{n-1} s_n$ получается строка, содержащая те же символы, но отсортированные в порядке неубывания их номеров по алфавиту. Например, из строки `hello` получится строка `ehllo`.

Задана строка и последовательность операций над некоторыми ее подстроками. Необходимо определить результат применения этих операций.

Формат входного файла

Первая строка входного файла содержит исходную строку. Она не пуста, состоит только из строчных букв латинского алфавита. Ее длина m не превышает 200 символов. Вторая строка входного файла содержит количество операций n ($1 \le n \le 200$).

Каждая из последующих n строк содержит описание одной операции. Описание операции имеет формат `OP L R`, где `OP` — символ, обозначающий операцию, `L` — позиция первого символа подстроки, к которой применяется операция, `R` — позиция ее последнего символа ($1 \le L \le R \le m$). Если до применения операции обрабатываемая строка имела вид $s_1 s_2 \ldots s_m$, то после применения операции она будет иметь вид $s_1 \ldots s_{L-1} OP(s_L \ldots s_R) s_{R+1} \ldots s_m$, где $OP(s_L \ldots s_R)$ — результат применения описываемой операции к строке $s_L \ldots s_R$.

Формат выходного файла

В выходной файл выведите результат применения к исходной строке всех перечисленных во входном файле операций.

Примеры

morpher.in	morpher.out
helloworld 2 S 1 5 R 6 10	ehllodlrow

Задача E. Оптимизация сепарабельной функции

Имя входного файла:	`optimize.in`
Имя выходного файла:	`optimize.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Функцию n переменных $f(x_1, x_2, \ldots, x_n)$ назовем *сепарабельной*, если она представима как сумма следующего вида $f(x_1, x_2, \ldots, x_n) = f_1(x_1) + f_2(x_2) + \ldots + f_n(x_n)$, где каждая f_i зависит только от x_i.

В рассматриваемой задаче каждая из функций f_i является многочленом второй степени от x_i.

Задача минимизации заданной функции на гиперпараллелепипеде формулируется следующим образом: «Задана функция $f(x_1, x_2, \ldots, x_n)$ и множество ограничений $a_1 \le x_1 \le b_1$, $a_2 \le x_2 \le b_2$, …, $a_n \le x_n \le b_n$. Необходимо найти точку $(x_1^*, x_2^*, \ldots, x_n^*)$ такую, что она удовлетворяет всем ограничениям, и значение функции в ней минимальное среди всех возможных».

Ваша задача состоит в том, чтобы решить задачу минимизации на гиперпараллелепипеде для заданной сепарабельной функции, состоящей из многочленов второй степени.

Формат входного файла

Первая строка входного файла содержит целое число n ($1 \le n \le 1000$). Каждая из последующих n строк содержит по пять целых чисел: a_i, b_i, p_i, q_i, r_i — они соответствуют ограничению $a_i \le x_i \le b_i$ и функции $f_i(x_i) = p_i \cdot x_i^2 + q_i \cdot x_i + r_i$ ($-1000 \le a_i \le b_i \le 1000$, $-10 \le p_i, q_i, r_i \le 10$, $p_i \ne 0$).

Формат выходного файла

В первой строке выведите минимальное возможное значение $f(x_1^*, x_2^*, \ldots, x_n^*)$. Во второй строке выведите x_1^*, x_2^*, …, x_n^*, на которых достигается указанное минимальное значение.

Ответ будет считаться правильным, если найденное минимальное значение отличается от правильного не более, чем на 10^{-3}.

Примеры

optimize.in	optimize.out
2 1 2 3 4 5 -1 1 1 2 3	14.0 1.0 -1.0
1 -1 1 1 0 0	0.0 0.0

Задача F. Декодирование префиксных кодов

Имя входного файла:	`prefix.in`
Имя выходного файла:	`prefix.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Кодом называют сопоставление $c : \Sigma \to \Gamma^*$ каждому символу некоторого алфавита Σ строк над некоторым (возможно другим) алфавитом Γ. В этой задаче Σ состоит из маленьких букв латинского алфавита, а $\Gamma = \{0, 1\}$.

Код называют *префиксным*, если ни одно кодовое слово не является префиксом другого кодового слова. Например, код $c(\text{‘a’}) = 00$, $c(\text{‘b’}) = 01$, $c(\text{‘c’}) = 1$ является префиксным, а код $c(\text{‘a’}) = 0$, $c(\text{‘b’}) = 01$ — нет.

Вам задан текст и строка, которая получается в результате кодирования этого текста некоторым префиксным кодом. Вам требуется восстановить код, который был использован для кодирования текста. Если возможных вариантов несколько, выведите любой.

Формат входного файла

Первая строка входного файла содержит заданный текст. Его длина не превышает 1000. Он состоит только из букв ‘`a`’–‘`z`’ латинского алфавита. Всего в тексте используется не более 10 различных букв.

Вторая строка содержит закодированную версию заданного текста. Гарантируется, что она была получена из текста кодированием его с использованием некоторого префиксного кода, длина каждого кодового слова в котором не превышает 10.

Формат выходного файла

Выведите в выходной файл код, который мог быть использован для кодирования текста. Количество строк в выводе должно совпадать с количеством различных символов в заданном тексте. Каждая строка должна содержать символ, после которого должен следовать пробел, а затем кодовое слово для этого символа.

Пример

prefix.in	prefix.out
hello 0100111000	e 001 h 01 l 1 o 000

Задача G. Гонки на лодках

Имя входного файла:	race.in
Имя выходного файла:	race.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Петя собирается участвовать в соревнованиях гоночных лодок. Гонка будет происходить в длинном узком канале. Канал проложен с запада на восток, берега канала имеют вид ломаной.

Введем систему координат таким образом, чтобы западный конец канала имел x-координату равную 0, а восточный конец канала — x-координату равную l. Ломаная, описывающая южный берег канала, имеет вершины в точках $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, где $0 = x_0 < x_1 < \ldots < x_n = l$. Северный берег канала имеет аналогичную форму, но находится на w к северу, таким образом он описывается ломаной с координатами вершин $(x_0, y_0 + w), (x_1, y_1 + w), \ldots, (x_n, y_n + w)$.

Лодка Пети может начать гонку в любой точке на стартовой прямой (отрезок $(0, y_0) - (0, y_0 + w)$) и закончить гонку в любой точке финишной прямой (отрезок $(l, y_n) - (l, y_n + w)$). Лодка настолько мала, что можно считать ее точкой. Когда лодка перемещается, она может проходить сколь угодно близко к берегам канала.

Чтобы увеличить свои шансы на победу, Петя хочет, чтобы кратчайший путь от стартовой до финишной позиции был как можно меньше. Помогите ему выяснить длину этого пути.

Формат входного файла

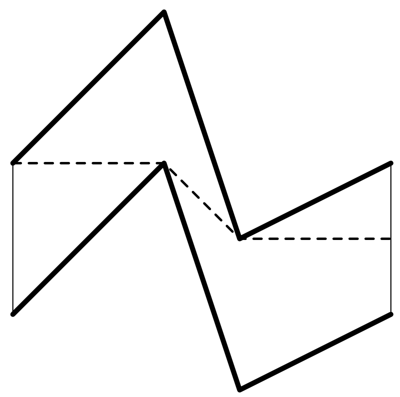
Первая строка входного файла содержит n ($1 \le n \le 100$). Следующие $n + 1$ строка содержит пары целых чисел (x_i, y_i) каждая и описывают южный берег канала ($0 = x_0 < x_1 < \ldots < x_n \le 10^4$, $|y_i| \le 10^4$). Последняя строка входного файла содержит целое число w ($1 \le w \le 10^4$).

Формат выходного файла

Выведите одно вещественное число — длину кратчайшего пути через канал от стартового прямой до финишной. Ответ должен отличаться от правильного не более чем на 10^{-6}.

Пример

race.in	race.out
3 0 0 2 2 3 -1 5 0 2	5.41421356237309505



Задача H. Космические путешествия

Имя входного файла:	`space.in`
Имя выходного файла:	`space.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

3141 год. Космические путешествия с использованием подпространственных телепортирующих туннелей уже давно не новинка. Однако еще не во всех частях Межгалактической Федерации Землян сеть таких туннелей достаточно развита.

В одной звездной системе n планет соединены туннелями так, что от каждой планеты можно добраться до каждой единственным способом, двигаясь только по туннелям. При этом перемещаться по гиперпространственному туннелю разрешается в обе стороны.

На каждой планете установлена подстанция, которая обеспечивает работу всех туннелей, которыми эта планета соединена с другими. Если эта подстанция перестает работать (например, из-за неполадок или из-за того, что ее закрывают на профилактический ремонт), то все туннели, одним из концов которых является эта планета, перестают работать. Вследствие этого для некоторых других планет может исчезнуть возможность добраться от одной до другой. Будем называть планеты, подстанции которых обладают описанным свойством, *важными*.

Поясним более формально. Планета u называется *важной*, если после того, как туннели, одним из концов которых является u, перестанут работать, появятся хотя бы две такие планеты v и w, что v нельзя добраться до w по оставшимся туннелям.

Задана схема гиперпространственных туннелей в рассматриваемой звездной системе. Ваша задача — написать программу, которая вычисляет количество важных планет в этой звездной системе.

Формат входного файла

Первая строка входного файла содержит n — количество ($1 \le n \le 10^5$) планет в звездной системе. Далее следуют $(n-1)$ строк, каждая из которых описывает один туннель и содержит два числа: u и v — номера планет, соединенных соответствующим туннелем ($1 \le u, v \le n$, $u \ne v$).

Планеты занумерованы натуральными числами от 1 до n.

Формат выходного файла

В выходной файл выведите ответ на задачу — количество важных планет.

Примеры

space.in	space.out
4 1 2 1 3 1 4	1
5 1 2 2 3 2 4 4 5	2