1. Такси

*Наши люди до метро на такси не ездят!*

После затянувшегося совещания директор фирмы решил заказать такси, чтобы развезти сотрудников по домам. Он заказал *N*машин — ровно столько, сколько у него сотрудников. Однако когда они подъехали, оказалось, что у каждого водителя такси свой тариф за 1 километр.

Директор знает, какому сотруднику сколько километров от работы до дома (к сожалению, все сотрудники живут в разных направлениях, поэтому нельзя отправить двух сотрудников на одной машине). Теперь директор хочет определить, какой из сотрудников на каком такси должен поехать домой, чтобы суммарные затраты на такси (а их несет фирма) были минимальны.

**Входные данные**

Сначала во входном файле записано натуральное число *N* (1≤*N*≤1000) — количество сотрудников компании (совпадающее с количеством вызванных машин такси). Далее записано *N* чисел, задающих расстояния в километрах от работы до домов сотрудников компании (первое число — для первого сотрудника, второе — для второго и т.д.). Все расстояния — положительные целые числа, не превышающие 1000. Далее записано еще *N* чисел — тарифы за проезд одного километра в такси (первое число — в первой машине такси, второе — во второй и т.д.). Тарифы выражаются положительными целыми числами, не превышающими 10000.

**Выходные данные**

В выходной файл выведите *N*чисел. Первое число — номер такси, в которое должен сесть первый сотрудник, второе число — номер такси, в которое должен сесть второй и т.д., чтобы суммарные затраты на такси были минимальны. Если вариантов рассадки сотрудников, при которых затраты минимальны, несколько, выведите любой из них.

**Примеры**

**входные данные**

3

10 20 30

50 20 30

**выходные данные**

1 3 2

**входные данные**

5

10 20 1 30 30

3 3 3 2 3

**выходные данные**

5 1 3 2 4

1. Химические реакции

Билл преподаёт химию в школе, он подготовил несколько тестов для учеников. Каждый тест состоит из химической формулы и нескольких возможных результатов реакции. Среди этих результатов ученики должны выбрать правильный. Билл хочет убедиться в том, что, вводя свои тесты в компьютер, он не допустил опечаток, благодаря которым ученики могли бы отбросить неверные ответы, просто подсчитав число химических элементов в левой и правой частях уравнения (в правильном уравнении химической реакции должно соблюдаться равенство).

Ваша задача - написать программу, которая поможет Биллу. Программа должна прочитать описание теста, состоящее из заданной левой части уравнения и нескольких возможных правых частей, и определить, равно ли количество химических элементов в каждой предложенной правой части уравнения количеству химических элементов в заданной левой части.

Билл формализовал задачу. И левая, и правая части уравнения представлены строкой символов без пробелов, состоящей из одной или более химических последовательностей, разделённых знаком плюс. Каждая последовательность имеет необязательный предшествующий целый множитель, относящийся ко всей последовательности, и несколько элементов. Каждый элемент может сопровождаться необязательным целым множителем, относящимся к нему. Элемент в этом уравнении может быть или отдельным химическим элементом, или целой последовательностью в круглых скобках. Каждый отдельный химический элемент представлен или одной прописной буквой, или прописной буквой, сопровождаемой строчной.

Ещё более формально, используя нотацию, аналогичную форме Бэкуса-Наура, можно написать:

* <формула> ::= [<число>] <последовательность> {"+" [<число>] <последовательность>}
* <последовательность> ::= <элемент> [<число>] {<элемент> [<число>]}
* <элемент> ::= <химический элемент> | "(" <последовательность> ")"
* <химический элемент> ::= <прописная буква> [<строчная буква>]
* <прописная буква> ::= "A".."Z"
* <строчная буква> ::= "a".."z"
* <число> ::= "1".."9" {"0".."9"}

Будем говорить, что каждый отдельный химический элемент встречается в формуле всего *X* раз, если *X* - сумма всех различных вхождений этого химического элемента, умноженных на все числа, относящиеся к ним. Например, в формуле C2H5OH+3O2+3(SiO2)

* C встречается всего 2 раза;
* H встречается всего 6 раз (5 + 1);
* O встречается всего 13 раз; (1 + 3 \* 2 + 3 \* 2);
* Si встречается всего 3 раза.

Все множители в формулах - целые числа не меньше 2, если заданы явно, или равны 1 - по умолчанию.

**Входные данные**

В первой строке находится формула - левая часть уравнения, во второй - одно число *N* - количество рассматриваемых правых частей, в каждой из следующих *N* строк - одна формула - предлагаемая правая часть уравнения.

Ограничения: 1 <= *N* <= 10, длина формулы не превосходит 100 символов, каждый отдельный химический элемент встречается всего не более 10 000 раз в каждой формуле.

**Выходные данные**

Для каждой из *N* заданных правых частей выведите одну строку вида

<формула левой части>==<формула правой части>

если общее количество вхождений каждого отдельного химического элемента в левую часть равно общему числу вхождений этого химического элемента в правую часть. В противном случае выведите:

<формула левой части>!=<формула правой части>

Здесь <формула левой части> должна быть замещена посимвольной копией формулы левой части, как она дана в первой строке входного файла, а <формула правой части> - замещена точной копией формулы правой части, как она дана во входном файле. В строках не должно быть пробелов.

**Примеры**

**входные данные**

C2H5OH+3O2+3(SiO2)

7

2CO2+3H2O+3SiO2

2C+6H+13O+3Si

99C2H5OH+3SiO2

3SiO4+C2H5OH

C2H5OH+3O2+3(SiO2)+Ge

3(Si(O)2)+2CO+3H2O+O2

2CO+3H2O+3O2+3Si

**выходные данные**

C2H5OH+3O2+3(SiO2)==2CO2+3H2O+3SiO2

C2H5OH+3O2+3(SiO2)==2C+6H+13O+3Si

C2H5OH+3O2+3(SiO2)!=99C2H5OH+3SiO2

C2H5OH+3O2+3(SiO2)==3SiO4+C2H5OH

C2H5OH+3O2+3(SiO2)!=C2H5OH+3O2+3(SiO2)+Ge

C2H5OH+3O2+3(SiO2)==3(Si(O)2)+2CO+3H2O+O2

C2H5OH+3O2+3(SiO2)!=2CO+3H2O+3O2+3Si

1. Финал

Ежегодно в Санкт-Петербурге, Барнауле и некоторых городах ближнего зарубежья проходят соревнования по программированию. Эти соревнования проходят в рамках студенческого чемпионата мира по программированию, организованного одной из самых авторитетных ассоциаций АСМ (Association for Computing Machinery). На этих соревнованиях проходит отбор команд с Северо-Восточного Европейского Региона NЕЕRС (North-Eastern European Regional Contest). Ежегодно перед организаторами соревнований встает проблема определения команд, которые будут приглашены к участию в финале чемпионата мира по программированию. По новым правилам в финал проходят не более *N*команд, представляющих NEERC. Кроме этого, от одного вуза не может проходить более чем *k*команд. При этот из всех таких множеств выбирается то, в котором сумма мест занятых этими командами в полуфинальных соревнованиях минимальная возможная. Ваша задача по итоговому протоколу полуфинальных соревнований и числам *N*и *k*определить, какие команды будут приглашены к участию в финале чемпионата мира.

**Входные данные**

В первой сроке входного файла находится три натуральных числа *Р*(1 ≤ *P*≤ 100000) — количество команд, принявших участие в полуфинале, *N*(1 ≤ *N*≤ *P*) и *k*(1 ≤ *k*≤ *P*) . В следующих *P*строках, по одному в строке перечислены названия университетов, команды которых заняли соответствующие места. Название университета содержит строчные и прописные латинские буквы и пробелы. Длина названия университета не превышает 30 символов. В следующей строке перечислены номера команд соответствующих университетов. Таким образам если название университета записано в *i*-той строке (2 ≤ *i*≤ *P*+ 1) , то эта команда заняла *i*- 1 место на полуфинале и имеет номер, записанный на *i*- 1 месте в *P*+ 2 строке.

**Выходные данные**

В выходной файл выведите названия команд, приглашенных к участию в финале чемпионата мира по программированию, упорядоченных по месту, занятому на полуфинале. В качестве названия команды выведите название университета и через пробел #номер команды.

**Примеры**

**входные данные**

9 5 2

Fantasy University

Crazy University

Fantasy University

Fantasy University

Very Good U

Good U

Very Good U

Crazy University

Good U

1 1 2 3 2 1 1 2 2

**выходные данные**

Fantasy University #1

Crazy University #1

Fantasy University #2

Very Good U #2

Good U #1

1. Партии

ограничение по времени на тест

1 секунда

ограничение по памяти на тест

256 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

В стране Байтландии есть *n* городов, соединенных дорогами. Каждая дорога двухсторонняя и имеет некоторую длину. Также, в стране есть две партии — одна выступает за то, что вилку необходимо держать во время еды в левой руке, другая — за то, что вилку необходимо держать в правой руке. В каждый момент времени каждый из городов поддерживает только одну партию. Вот только время от времени в городах происходят перевороты и в одном из городов поддерживаемая партия меняется на противоположную.

Правительство Байтландии поручило Вам написать программу, которая бы в каждый момент времени оценивала стабильность политической ситуации в стране. Согласно математической модели, разработанной ведущими учеными Байтландии, стабильность зависит от того, насколько близко находятся города, поддерживающие одну и ту же партию, поскольку чем ближе они, чем больше вероятность того, что они могут объединиться в коалицию.

Поэтому программа, написанная Вами, должна по информации о всех городах, дорогах и происходящих переворотах находить, после каждого переворота, два города, поддерживающих одну партию, и находящихся на наименьшем расстоянии друг от друга.

**Входные данные**

Первая строка входного файла содержит три целых числа *n*, *m* и *k* (1 ≤ *n*, *m*, *k* ≤ 100000) — количество городов, дорог и переворотов соответственно.

Вторая строка содержит строку *s* длиной *n*, описывающую политическую ситуацию в стране на момент начала наблюдения. Символ L на *i*-ой позиции означает, что город номер *i* поддерживает партию, выступающую за то, что вилку необходимо держать в левой руке, символ R — в правой.

Следующие *m* строк содержат по три целых числа *ai*, *bi* и *li* (1 ≤ *ai*, *bi* ≤ *n*, *ai* ≠ *bi*, 1 ≤ *li* ≤ 109) — номера городов, которые соединены *i*-ой дорогой и ее длина. Каждая пара городов соединена не более, чем одной дорогой.

В последней строке содержится *k* чисел --- номера городов *cj* (1 ≤ *cj* ≤ *n*), в которых происходили перевороты в порядке их происшествия. Поскольку все города достаточно консервативны, то в каждом городе произошло не более одного переворота, то есть все *cj* различны.

**Выходные данные**

В выходной файл выведите *k* + 1 отчет о ближайших городах, которые могут объединиться в коалицию. Первое описание должно соответствовать началу наблюдения, каждое следующее — после очередного переворота в одном из городов.

Каждое описание должно быть выведено на отдельной строке и содержать три целых числа *di*, *xi* и *yi* — минимальное расстояния между городами, поддерживающими одну партию и номера этих городов, соответственно. Если таких пар несколько, выведите любую. Гарантируется, что хотя бы одна такая пара всегда существует.

**Подгруппа 1.**

*n* ≤ 100. Решение оценивается в 30 баллов.

**Подгруппа 2.**

*n* ≤ 2000, Решение оценивается в 30 баллов.

**Подгруппа 3.**

Дополнительные ограничения отсутствуют. Решение оценивается в 40 баллов.

**Примеры**

**входные данные**

5 6 4

LRLRL

1 4 1

2 3 2

3 4 3

4 5 4

2 5 5

2 4 6

1 4 3 5

**выходные данные**

4 1 3

1 1 4

3 3 4

2 2 3

2 2 3