Задача A База данных

Максимальное время работы на одном тесте:	4 секунды
Максимальный объем используемой памяти:	64 мегабайта

Ценные бумаги на фондовом рынке характеризуются множеством параметров. У них есть цена и ликвидность, также оценивать динамичность изменения цены, среднюю прибыльность, потенциал роста прибыльности и др. показатели. Аналитики трейдовой компании "WebMarket" ввели специальный показатель *надежности ценной бумаги* и научились эффективно его оценивать. Большое значение этого показателя соответствует малому риску покупки ценной бумаги. Но с ростом надежности обычно падает среднее оцениваемое значение прибыльности.

Для своих клиентов, играющих на рынке ценных бумаг, компания "WebMarket" решила открыть значения этого показателя и более того, автоматизировать покупку ценных бумаг с заданным порядковым номером по значению надежности. Аналитики проанализировали идею, и решили, что наличие такого функционала будет способствовать привлечению новых клиентов на рынок "WebMarket", повышению объемов сделок, а значит, и повышению прибылей "WebMarket". Важно также отметить, что торговля на базе этого показателя может позитивно сказаться на российском фондовом рынке и сделать его более здоровым. Алгоритмы оценки надежности уже написаны, средства выделены, необходимая реклама проведена. Осталось только написать сам код.

Ценные бумаги в базе данных имеют три атрибута:

- `code` — непустая строка латинских символов и цифр длины 30 или меньше
- `id` — целочисленный идентификатор (начиная с 0)
- `reliability` — целое число из диапазона $[-2^{31}, 2^{31})$.

Каждой новой ЦБ выдается следующий по порядку `id` и значение её надёжности устанавливается в 0. Если ценная бумага отзывается с рынка, ее `id` для новых бумаг не используется.

База данных получает запросы, которые позволяют вводить новые ЦБ на рынок, получать текущую информацию о ЦБ, отзывать ЦБ с рынка, менять значение надежности у ЦБ и находить ЦБ, которая стоит на `n`-м месте, если упорядочить ЦБ по убыванию надежности, а при одинаковых значениях по возрастанию идентификатора.

При добавлении ЦБ с кодом, который раньше встречался, но соответствующая ЦБ была отозвана с рынка, ей назначается новый идентификатор.

Входные и выходные данные

Первая строка входа содержит число запросов `N`. ($1 \le N \le 100000$). Затем идут `N` строк, каждая из которых содержит один запрос. Запросы бывают 4 типов:

- `ISSUE code`: добавить новую ЦБ `code` в базу данных, или вывести информацию о ЦБ, если она уже есть в базе
- если ЦБ `code` существует, то вывести `EXISTS id reliability`
- если ЦБ `code` не существует, то вывести `CREATED id 0`
- `DELETE code`: удалить ЦБ `code` из базы
- если ЦБ `code` существует, то вывести `OK id reliability`
- если ЦБ `code` не существует, то вывести `BAD REQUEST`
- `RELIABILITY code reliability`: увеличить количество очков ЦБ, гарантируется, что значение останется в диапазоне $[-2^{31}, 2^{31})$.
- если ЦБ `code` существует, то вывести `OK id new_reliability`
- если ЦБ `code` не существует, то вывести `BAD REQUEST`
- `FIND n`: найти `n`-ю ЦБ (начиная с 0)
- если база не пуста, найти ЦБ, которая имела бы порядковый номер `n` (нумерация начинается с 0), если все ЦБ упорядочить сначала по убыванию баллов, а группы ЦБ с одинаковыми баллами упорядочить по времени их добавления (то есть по возрастанию id); а если `n` больше, чем количество ЦБ в базе, то найти последнюю ЦБ в этом порядке; для обоих случаев вывести
  OK code id reliability

- если база пуста, то вывести `EMPTY`

Таким образом, на каждый запрос на входе нужно вывести одну строку с результатом.

Пример

Входные данные	Выходные данные
17 ISSUE aaa FIND 10 ISSUE bbb ISSUE ccc RELIABILITY aaa 10 RELIABILITY bbb 30 RELIABILITY ccc 20 RELIABILITY xxx 20 FIND 1 FIND 2 FIND 0 ISSUE eee ISSUE fff FIND 3 FIND 111 DELETE bbb FIND 0	CREATED 0 0 OK aaa 0 0 CREATED 1 0 CREATED 2 0 OK 0 10 OK 1 30 OK 2 20 BAD REQUEST OK ccc 2 20 OK aaa 0 10 OK bbb 1 30 CREATED 3 0 CREATED 4 0 OK eee 3 0 OK fff 4 0 OK 1 30 OK ccc 2 20

Задача B Форматирование документа

Максимальное время работы на одном тесте:	2 секунды
Максимальный объем используемой памяти:	64 мегабайта

Вася пишет новую версию своего офисного пакета «Closed Office». Недавно он начал работу над редактором «Dword», входящим в состав пакета.

Последняя проблема, с которой столкнулся Вася — размещение рисунков в документе. Он никак не может добиться стабильного отображения рисунков в тех местах, в которые он их помещает. Окончательно отчаявшись написать соответствующий модуль самостоятельно, Вася решил обратиться за помощью к вам.

Напишите программу, которая будет осуществлять размещение документа на странице.

Документ в формате редактора <<Dword>> представляет собой последовательность абзацев. Каждый абзац представляет собой последовательность элементов — слов и рисунков. Элементы одного абзаца разделены пробелами и/или переводом строки. Абзацы разделены пустой строкой. Строка, состоящая только из пробелов, считается пустой.

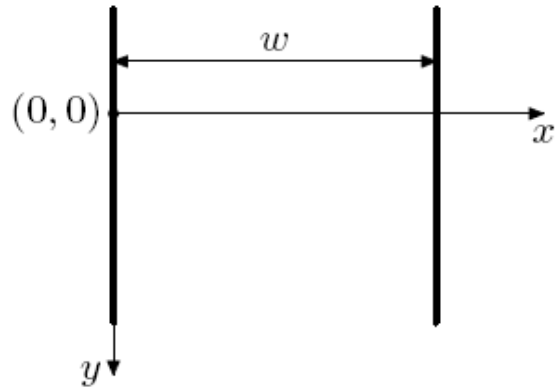
Слово — это последовательность символов, состоящая из букв латинского алфавита, цифр, и знаков препинания: «.», «,», «:», «;», «!», «?», «-», «'».

Рисунок описывается следующим образом: «(image *параметры рисунка*)». Каждый параметр рисунка имеет вид «имя = *значение*». Параметры рисунка разделены пробелами и/или переводом строки.

У каждого рисунка обязательно есть следующие параметры:

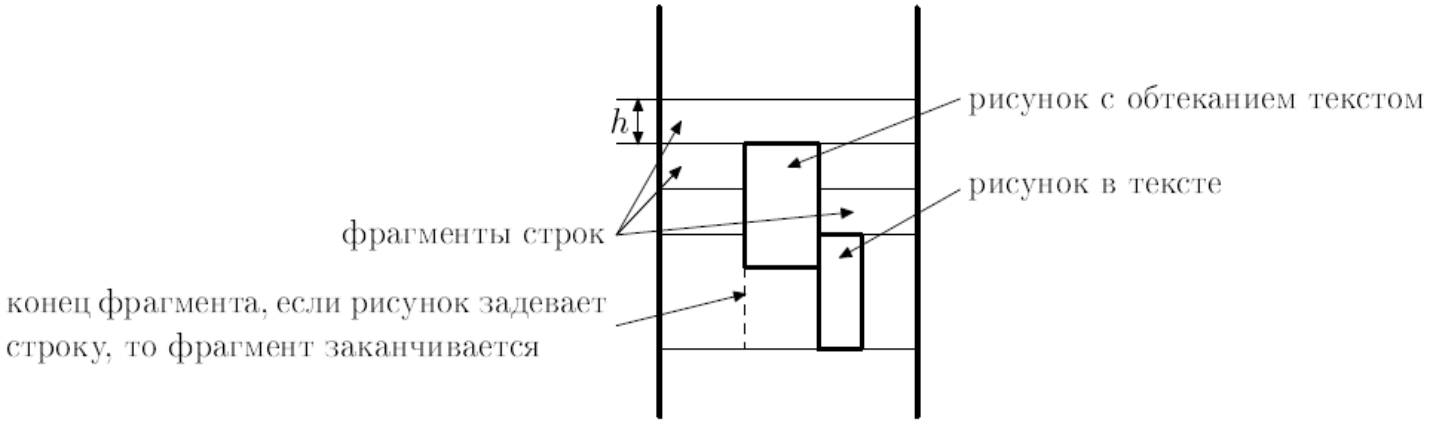
Параметр	**Описание**
`width`	Целое положительное число — ширина рисунка в пикселях
`height`	Целое положительное число --- высота рисунка в пикселях
`layout`	Одно из следующих значений: `embedded` (в тексте), `surrounded` (обтекание текстом), `floating` (свободное) — описывает расположение рисунка относительно текста

Документ размещается на бесконечной вверх и вниз странице шириной w пикселей (разбиение на конечные по высоте страницы планируется в следующей версии редактора). Одна из точек на левой границе страницыусловно считается точкой с ординатой равной нулю. Ордината увеличивается вниз.



Размещение документа происходит следующим образом. Абзацы размещаются по очереди. Первый абзац размещается так, что его верхняя граница имеет ординату 0.

Абзац размещается следующим образом. Элементы располагаются по строкам. Каждая строка исходно имеет высоту h пикселей. В процессе размещения рисунков высота строк может увеличиваться, и строки могут разбиваться рисунками на фрагменты.



Слова размещаются следующим образом. Считается, что каждый символ имеет ширину с пикселей. Перед каждым словом, кроме первого во фрагменте, ставится пробел шириной также в с пикселей. Если

слово помещается в текущем фрагменте, то оно размещается на нем. Если слово не помещается в текущем фрагменте, то оно размещается в первом фрагменте текущей строки, расположенном правее текущего, в котором оно помещается. Если такого фрагмента нет, то начинается новая строка, и поиск подходящего фрагмента продолжается в ней. Слово всегда «прижимается» к верхней границе строки.

Размещение рисунка зависит от его расположения относительно текста.

Если расположение рисунка относительно текста установлено в «embedded», то он располагается так же как слово, за тем исключением, что его ширина равна ширине, указанной в параметрах рисунка. Кроме того, если высота рисунка больше текущей высоты строки, то она увеличивается до высоты рисунка (при этом верхняя граница строки не перемещается, а смещается вниз нижняя граница). Если рисунок типа «embedded» не первый элемент во фрагменте, то перед ним ставится пробел шириной с пикселей. Рисунки типа «embedded» также прижимаются к верхней границе строки.

Если расположение рисунка относительно текста установлено в «surrounded», то рисунок размещается следующим образом. Сначала аналогично находится первый фрагмент, в котором рисунок помещается по ширине. При этом перед рисунком этого типа не ставится пробел, даже если это не первый элемент во фрагменте.

После этого рисунок размещается следующим образом: верхний край рисунка совпадает с верхней границей строки, в которой находится найденный фрагмент, а сам рисунок продолжается вниз. При этом строки, через которые он проходит, разбиваются им на фрагменты.

Если расположение рисунка относительно текста установлено в «floating», то рисунок размещается поверх текста и других рисунков и никак с ними не взаимодействует. В этом случае у рисунка есть два дополнительных параметра: «dx» и «dy» - целые числа, задающие смещение в пикселях верхнего левого угла рисунка вправо и вниз, соответственно, относительно позиции, где находится верхний правый угол предыдущего слова или рисунка (или самой левой верхней точки первой строки абзаца, если рисунок — первый элемент абзаца).

Если при размещении рисунка таким образом он выходит за левую границу страницы, то он смещается вправо, так, чтобы его левый край совпадал с левой границей страницы. Аналогично, если рисунок выходит за правую границу страницы, то он смещается влево, чтобы его правый край совпадал с правой границей страницы.

Верхняя граница следующего абзаца совпадает с более низкой точкой из нижней границы последней строки и самой нижней границы рисунков типа «surrounded» предыдущего абзаца.

По заданным w, h, c и документу найдите координаты верхних левых углов всех рисунков в документе.

Входные данные

Первая строка входного файла содержит три целых числа: w, h и c ($1 \le w \le 1000$, $1 \le h \le 50$, $1 \le c \le w$).

Далее следует документ. Размер входного файла не превышает 1000 байт. Гарантируется, что ширина любого слова и любого рисунка не превышает w. Высота всех рисунков не превышает 1000. Относительное смещение всех рисунков типа «floating» не превышает 1000 по абсолютной величине.

Выходные данные

Выведите в выходной файл по два числа для каждого рисунка — координаты его верхнего левого угла. Выводите координаты рисунков в том порядке, в котором они встречаются во входном файле.

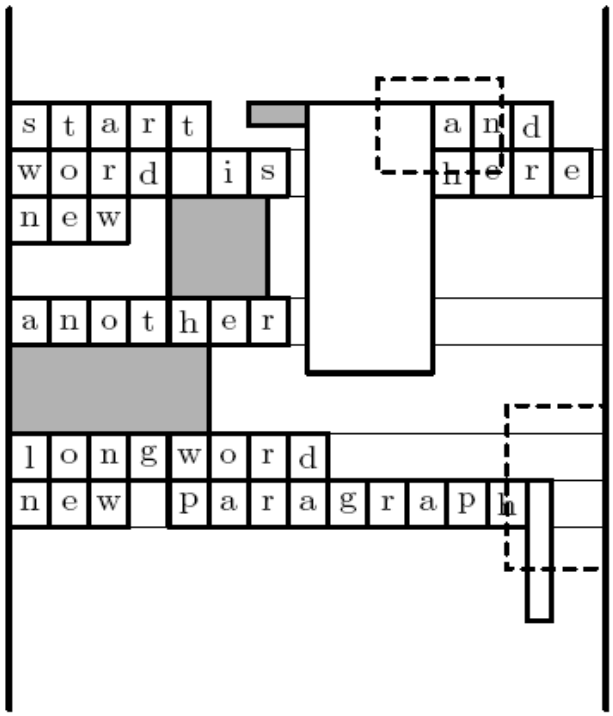
Пример

Входные данные
120 10 8 start (image layout=embedded width=12 height=5) (image layout=surrounded width=25 height=58) and word is (image layout=floating dx=18 dy=-15 width=25 height=20) here new (image layout=embedded width=20 height=22) another (image layout=embedded width=40 height=19) longword new paragraph (image layout=surrounded width=5 height=30)

(image layout=floating width=20 height=35 dx=50 dy=-16)

Выходные данные

```
48 0
60 0
74 -5
32 20
0 52
104 81
100 65
```



Задача C ДКА

Максимальное время работы на одном тесте:	2 секунды
Максимальный объем используемой памяти:	64 мегабайта

Напишите программу, моделирующую работу детерминированного конечного автомата (ДКА). Описание автомата и входная строка вводятся на стандартном потоке ввода. Результат работы автомата над данной строкой выводится на стандартный поток вывода.

Входные данные

Описание автомата задаётся в следующей форме. Сначала задаётся функция перехода автомата. Функция перехода задаётся в виде троек CUR CHAR NEW, где CUR — идентификатор исходного состояния — произвольная символьная строка, не содержащая пробельные символы. CHAR — символьная строка длиной ровно 1 символ. NEW — идентификатор целевого состояния — произвольная символьная строка, не содержащая пробельные символы. Элементы описания перехода могут отделятся друг от друга произвольным количеством пробельных символов. Описание функции перехода завершается строкой "END" в качестве идентификатора исходного состояния. Элементы CHAR и NEW отсутсвуют.

Далее перечисляются заключительные состояния автомата. Каждое состояние — это символьная строка. Список состояний завершается символьной строкой "END". Далее задаётся начальное состояние автомата — символьная строка. Затем задаётся проверяемое слово — символьная строка. Все элементы входного файла могут отделяться друг от друга произвольным количеством пробельных символов. Можете предполагать, что входные данные корректны, то есть удовлетворяют спецификации и действительно задают детерминированный конечный автомат.

Выходные данные

Результат работы автомата должен быть напечатан в следующем виде. Сначала напечатайте число 1, если данный автомат допускает данную цепочку, и 0 в противном случае. Затем напечатайте количество символов, прочитанных во входной цепочке к моменту принятия автоматом решения. Наконец, напечатайте идентификатор состояния, в котором в данный момент находился автомат.

Примеры

Входные данные	Выходные данные
A a A A b B B a C B b B C a C END B C END A aaabbbbba	1 9 C
A a A A b B B a C B b B C a C END B C END A abab	0 3 C

Задача D НКА

Максимальное время работы на одном тесте:	2 секунды
Максимальный объем используемой памяти:	64 мегабайта

Напишите программу, моделирующую работу недетерминированного конечного автомата (НКА). Описание автомата и входная строка вводятся на стандартном потоке ввода. Результат работы автомата над данной строкой выводится на стандартный поток вывода.

Входные данные

Формат входных данных полностью аналогичен формату входных данных предыдущего задания, за исключением того, что CHAR может быть равной строке "eps". В этом случае задаётся varepsilon — переход из состояния CUR в состояние NEW. Поскольку автомат недетерминированный, каждая пара CUR, CHAR может повторяться несколько раз.

Выходные данные

Результат работы автомата должен быть напечатан в следующем виде. Сначала напечатайте число 1, если данный автомат допускает данную цепочку, и 0 в противном случае. Затем напечатайте количество символов, прочитанных во входной цепочке к моменту принятия автоматом решения.

Пример

Входные данные	Выходные данные
A a A A b A A eps B B a C C a D C b D END D END A abaaab	1 6