Задача A. Менеджер памяти

Имя входного файла:	`memory.in`
Имя выходного файла:	`memory.out`
Ограничение по времени:	6 секунды
Ограничение по памяти:	512 мегабайт

Одно из главных нововведений новейшей операционной системы Indows 7 — новый менеджер памяти. Он работает с массивом длины N и позволяет выполнять три самые современные операции:

- `copy(a, b, l)` — скопировать отрезок длины $[a, a+l-1]$ в $[b, b+l-1]$
- `sum(l, r)` — посчитать сумму элементов массива на отрезке $[l, r]$
- `print(l, r)` — напечатать элементы с l по r, включительно

Вы являетесь разработчиком своей операционной системы, и Вы, безусловно, не можете обойтись без инновационных технологий. Вам необходимо реализовать точно такой же менеджер памяти.

Формат входного файла

Первая строка входного файла содержит целое число N ($1 \le N \le 1\,000\,000$) — размер массива, с которым будет работать Ваш менеджер памяти.

Во второй строке содержатся четыре числа $1 \le X_1, A, B, M \le 10^9 + 10$. С помощью них можно сгенерировать исходный массив чисел $X_1, X_2, \cdots, X_N$. $X_{i+1} = (A * X_i + B) \mod M$

Следующая строка входного файла содержит целое число K ($1 \le K \le 200\,000$) — количество запросов, которые необходимо выполнить Вашему менеджеру памяти.

Далее в K строках содержится описание запросов. Запросы заданы в формате:

- `cpy` a b l — для операции `copy`
- `sum` l r — для операции `sum` ($l \le r$)
- `out` l r — для операции `print` ($l \le r$)

Гарантируется, что суммарная длина запросов `print` не превышает 3 000. Также гарантируется, что все запросы корректны.

Формат выходного файла

Для каждого запроса *sum* или *print* выведите в выходной файл на отдельной строке результат запроса.

Примеры

memory.in	memory.out
6 1 4 5 7 7 out 1 6 cpy 1 3 2 out 1 6 sum 1 4 cpy 1 2 4 out 1 6 sum 1 6	1 2 6 1 2 6 1 2 1 2 2 6 6 1 1 2 1 2 6 13

Задача B. Самая дальняя

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

Даны N точек на плоскости, нужно уметь обрабатывать следующие запросы:

- `get a b` — возвращает максимум по всем точкам величины $a \cdot x + b \cdot y$.
- `add x y` — добавить точку в множество.

Формат входного файла

Число N ($1 \le N \le 10^5$) и N точек. Далее число M ($1 \le M \le 10^5$) — количество запросов и собственно запросы. Формат запросов можно посмотреть в примере. Все координаты точек и числа `a`, `b` — целые числа, по модулю не превосходящие 10^9.

Формат выходного файла

На каждый запрос вида `get` выведите одно целое число — максимум величины $a \cdot x + b \cdot y$.

Примеры

mostfar.in	mostfar.out
3 0 0 1 0 0 1 10 get 1 1 get -1 -1 get 1 -1 get -1 1 add 2 2 add -2 -2 get 1 1 get -1 -1 get 1 -1 get -1 1	1 0 1 1 4 4 1 1

Задача C. Ожерелья

Имя входного файла:	necklace.in
Имя выходного файла:	necklace.out
Ограничение по времени:	5 секунды
Ограничение по памяти:	256 мегабайт

Алиса: «У меня было самое красивое ожерелье. Слева направо, оно состояло из двух красных бусин, двух зеленых и еще одной красной».

Биатрисс быстро нашла, что ответить: «А мое было еще лучше. Оно выглядело почти как твое, если убрать две самые правые бусины и добавить вместо них две голубых».

Как только она закончила говорить, в обсуждение быстро вступила Каролина: «Оно почти как мое. Толлько у него есть еще одна желтая бусина слева».

Вы, наверное, уже не удивитесь, когда я вам скажу, что Доминика тоже не осталась в стороне. «Все ваши ожерелья такие скучные. Чтобы получить мое, надо взять ожерелье Биатрисс, убрать самую левую и самую правую бусину, и добавить две черных слева».

И это продолжалось, пока Заида не спросила: «Я немного запуталась. Какого цвета была самая левая бусина в ожерелье Евгении?» На этот вопрос никто не смог ответить.

Может быть, вы поможете девушкам?

Ваша программа должна уметь обрабатывать множество ожерелий. Ожерелье — это последовательность целых чисел от 0 до 1 000 000, упорядоченных слева направо. У каждого ожерелья есть номер. Изначально есть одно пустое ожерелье. Оно имеет номер 0. Далее вам поступает не более 1 000 001 запросов.

Запросы бывают следующих типов.

- `A` *id side color*. Этот запрос означает, что надо создать новое ожерелье из ожерелья с номером *id* путем добавления бусины цвета *color* со стороны *side*. Параметр *side* — это буква `L`, если надо добавить бусину слева, и `R`, если справа. Номер нового ожерелья на 1 больше самого большого из уже существующих номеров. Гарантируется, что ожерелье с номером *id* уже существует.
- `R` *id side*. Этот запрос означает, что надо создать новое ожерелье из ожерелья с номером *id* путем удаления бусины со стороны *side*. Сторона задается аналогично первому запросу. Номер нового ожерелья на 1 больше самого большого из уже существующих номеров. Гарантируется, что ожерелье с номером *id* уже существует и не является пустым.
- `Q` *id side*. В качестве ответа на этот запрос надо вывести одно число — цвет бусины со стороны *side* в ожерелье *id*. Гарантируется, что оно существет и не является пустым.
- `X`. Этот запрос означает, что надо завершить выполнение программы.

Формат входного файла

На ввод вашей программе подается последовательность запросов в описанном выше формате. Количество запросов не превосходит $10^6 + 1$.

Формат выходного файла

В ответ на каждый запрос типа `Q` выведите ответ на него в отдельной строке.

Примеры

necklace.in	necklace.out
A 0 L 1 A 1 L 2 Q 2 R R 2 R Q 3 R Q 2 R X	1 2 1

Note

Данную задачу надо решать online, персистентным деком. Остальные решения получат Ignored :)

Формально это означает, что ответ на каждый запрос должен быть выведен до считывания следующего запроса.

Задача D. Персистентная очередь

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

Реализуйте персистентную очередь.

Формат входного файла

Первая строка содержит количество действий n ($1 \le n \le 200\,000$). В строке номер $i+1$ содержится описание действия i:

- `1 t m` — добавить в конец очереди номер t ($0 \le t < i$) число m;
- `-1 t` — удалить из очереди номер t ($0 \le t < i$) первый элемент.

В результате действия i, описанного в строке $i+1$ создается очередь номер i. Изначально имеется пустая очередь с номером ноль.

Все числа во входном файле целые, и помещаются в знаковый 32-битный тип.

Формат выходного файла

Для каждой операции удаления выведите удаленный элемент на отдельной строке.

Примеры

queue.in	queue.out
10 1 0 1 1 1 2 1 2 3 1 2 4 -1 3 -1 5 -1 6 -1 4 -1 8 -1 9	1 2 3 1 2 4

Задача E. Персистентный стек

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

Реализуйте персистентный стек.

Формат входного файла

Первая строка содержит количество действий n ($1 \le n \le 200\,000$). В строке номер $i+1$ содержится описание действия i:.

- `t m` — добавить в конец стека номер t ($0 \le t < i$) число m ($0 < m \le 1000$);
- `t 0` — удалить последний элемент стека номер t ($0 \le t < i$). Гарантируется, что стек t не пустой.

В результате действия i, описанного в строке $i+1$, создается стек номер i. Изначально имеется пустой стек с номером ноль.

Все числа во входном файле целые.

Формат выходного файла

Для каждой операции удаления выведите удаленный элемент на отдельной строке.

Примеры

stack.in	stack.out
8 0 1 1 5 2 4 3 2 4 3 5 0 6 6 1 0	3 1

Задача F. Откат

Имя входного файла:	`rollback.in`
Имя выходного файла:	`rollback.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайта

Сергей работает системным администратором в очень крупной компании. Естественно, в круг его обязанностей входит резервное копирование информации, хранящейся на различных серверах и «откат» к предыдущей версии в случае возникновения проблем.

В данный момент Сергей борется с проблемой недостатка места для хранения информации для восстановления. Он решил перенести часть информации на новые сервера. К сожалению, если что-то случится во время переноса, он не сможет произвести откат, поэтому процедура переноса должна быть тщательно спланирована.

На данный момент у Сергея хранятся n точек восстановления различных серверов, пронумерованных от 1 до n. Точка восстановления с номером i позволяет произвести откат для сервера a_i. Сергей решил разбить перенос на этапы, при этом на каждом этапе в случае возникновения проблем будут доступны точки восстановления с номерами $l, l+1, \ldots, r$ для некоторых l и r.

Для того, чтобы спланировать перенос данных оптимальным образом, Сергею необходимо научиться отвечать на запросы: для заданного l, при каком минимальном r в процессе переноса будут доступны точки восстановления не менее чем k различных серверов.

Помогите Сергею.

Формат входного файла

Первая строка входного файла содержит два целых числа n и m, разделенные пробелами — количество точек восстановления и количество серверов ($1 \le n, m \le 100\,000$). Вторая строка содержит n целых чисел $a_1, a_2, \ldots, a_n$ — номера серверов, которым соответствуют точки восстановления ($1 \le a_i \le m$).

Третья строка входного файла содержит q — количество запросов, которые необходимо обработать ($1 \le q \le 100\,000$). В процессе обработки запросов необходимо поддерживать число p, исходно оно равно 0. Каждый запрос задается парой чисел x_i и y_i, используйте их для получения данных запроса следующим образом: $l_i = ((x_i + p) \bmod n) + 1$, $k_i = ((y_i + p) \bmod m) + 1$ ($1 \le l_i, x_i \le n$, $1 \le k_i, y_i \le m$). Пусть ответ на i-й запрос равен r. После выполнения этого запроса, следует присвоить p значение r.

Формат выходного файла

На каждый запрос выведите одно число — искомое минимальное r, либо 0, если такого r не существует.

Примеры

rollback.in	rollback.out
7 3 1 2 1 3 1 2 1 4 7 3 7 1 7 1 2 2	1 4 0 6

Задача G. Строки в дереве

Имя входного файла:	treestr.in
Имя выходного файла:	treestr.out
Ограничение по времени:	5 секунды
Ограничение по памяти:	256 мегабайт

Дано дерево. Дерево — это связный граф без циклов. На каждом ребре дерева написана строчная латинская буква. Между каждыми двумя вершинами существует ровно один простой путь, то есть путь по рёбрам дерева, проходящий через каждую вершину не более одного раза. Каждому пути соответствует строка, которая получается, если идти по этому пути и читать буквы на рёбрах в порядке следования. Путь можно проходить, начиная с любого его конца.

Также дана строка S. Соответствует ли она какому-либо простому пути в данном дереве?

Длина строки и размер дерева не превышают $3 \cdot 10^5$.

Формат входного файла

Первая строка содержит заданную строку s. Следующая строка содержит количество вершин в дереве n. Следующие $n-1$ строк описываются ребра деревав виде u, v, c, где u и v — вершины дерева, а c — символ, написанный на ребре.

Формат выходного файла

В первой строке выведите YES, если такой путь существует, и NO в противном случае.

Если путь существует, то выведите пару вершин, путь между которыми образует заданную строку.

Примеры

treestr.in	treestr.out
abc 4 1 2 c 4 3 a 2 4 b	YES 1 3
abc 1	NO
zy 6 1 2 x 1 3 y 1 4 z 3 5 a 4 6 b	YES 4 3

Note

В первом примере строка abc соответствует пути 3–4–2–1.

Во втором примере в дереве нет ни одного ребра.

В третьем примере строка zy соответствует пути 3–1–4.

Задача H. 2D-переворот

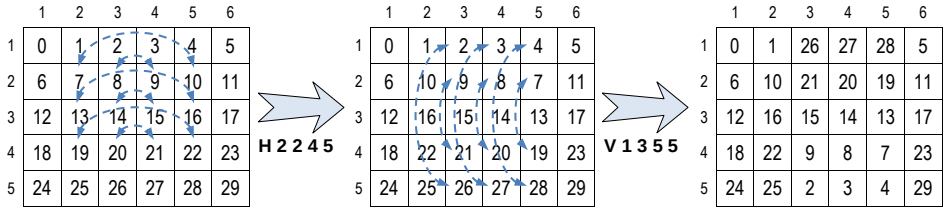
Имя входного файла:	reverse2d.in
Имя выходного файла:	reverse2d.out
Ограничение по времени:	5 секунды
Ограничение по памяти:	256 мегабайт

You are given a matrix A containing N rows numbered from 1 to N and M columns numbered from 1 to M. Initially the element in row i, column j is $A_{i,j} = (i-1) \cdot M + j - 1$.

Someone has done several transformations to the matrix. Each transformation was either a horizontal reverse or a vertical reverse. Each reverse is described by four integers X_1, Y_1, X_2 and Y_2 such that $1 \le X_1 \le X_2 \le N$ and $1 \le Y_1 \le Y_2 \le M$.

After the horizontal reverse the element in row i, column j becomes equal to $A'_{i,j} = A_{i,Y_2-j+Y_1}$ iff $X_1 \le i \le X_2$ and $Y_1 \le j \le Y_2$, otherwise it remains the same ($A'_{i,j} = A_{i,j}$). After the vertical reverse the element in row i, column j becomes equal to $A'_{i,j} = A_{X_2-i+X_1,j}$ iff $X_1 \le i \le X_2$ and $Y_1 \le j \le Y_2$, otherwise it remains the same ($A'_{i,j} = A_{i,j}$).

Below you can see an example of applying a horizontal reverse and then a vertical reverse.



You should write a program that, given N, M and a sequence of operations, will be able to find the resulting matrix.

Формат входного файла

The first line contains two space-separated integers N and M ($1 \le N, M \le 2000$). The next line contains an integer K ($1 \le K \le 2000$) — the number of operations. Each of the next K lines describes one operation. The first character of the line stands for the type of the operation ('`H`' for horizontal reverse or '`V`' for vertical reverse) after which there are four integers X_1, Y_1, X_2 and Y_2 — description of the selected rectangular range. Operations are given in order of their applying.

Формат выходного файла

The first line should contain N space-separated integers corresponding to the sum of the elements in the first, second, ..., N-th row of the matrix after applying all the operations. Similarly, the second line should contain M space-separated integers — the sum of the elements in the first, second, ..., M-th column of the resulting matrix.

Примеры

reverse2d.in	reverse2d.out
5 6 2 H 2 2 4 5 V 1 3 5 5	87 87 87 87 87 60 74 73 72 71 85
5 4 6 H 1 1 1 1 V 4 4 5 4 H 1 2 4 4 V 1 2 5 3 H 3 1 5 4 H 2 3 3 4	36 42 38 38 36 41 54 41 54