Задача A. Двоичное дерево поиска

Имя входного файла:	`bst.in`
Имя выходного файла:	`bst.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Реализуйте сбалансированное двоичное дерево поиска.

Формат входных данных

Входной файл содержит описание операций с деревом, их количество не превышает 100000. В каждой строке находится одна из следующих операций:

- `insert` x — добавить в дерево ключ x.
- `delete` x — удалить из дерева ключ x. Если ключа x в дереве нет, то ничего делать не надо.
- `exists` x — если ключ x есть в дереве, выведите «`true`», иначе «`false`»
- `next` x — выведите минимальный элемент в дереве, строго больший x, или «`none`», если такого нет.
- `prev` x — выведите максимальный элемент в дереве, строго меньший x, или «`none`», если такого нет.

Все числа во входном файле целые и по модулю не превышают 10^9.

Формат выходных данных

Выведите последовательно результат выполнения всех операций `exists`, `next`, `prev`. Следуйте формату выходного файла из примера.

Примеры

bst.in	bst.out
insert 2 insert 5 insert 3 exists 2 exists 4 next 4 prev 4 delete 5 next 4 prev 4	true false 5 3 none 3

Задача B. Вперёд!

Имя входного файла:	`movetofront.in`
Имя выходного файла:	`movetofront.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Капрал Дукар любит раздавать приказы своей роте. Самый любимый его приказ — «Вперёд!». Капрал строит солдат в ряд и отдаёт некоторое количество приказов, каждый из которых звучит так: «Рядовые с l_i по l_j — вперёд!»

Перед тем, как Дукар отдал первый приказ, солдаты были пронумерованы от 1 до n слева направо. Услышав приказ «Рядовые с l_i по l_j — вперёд!», солдаты, стоящие на местах с l_i по l_j включительно, продвигаются в начало ряда в том же порядке, в котором были.

Например, если в какой-то момент солдаты стоят в порядке $2, 3, 6, 1, 5, 4$, то после приказа «Рядовые с 2 по 4 — вперёд!», порядок будет таким: $3, 6, 1, 2, 5, 4$. А если потом Капрал вышлет вперёд солдат с 3 по 4, то порядок будет уже таким: $1, 2, 3, 6, 5, 4$.

Вам дана последовательность приказов Капрала. Найдите порядок, в котором будут стоять солдаты после исполнения всех приказов.

Формат входных данных

В первой строке входного файла указаны числа n и m ($2 \leqslant n \leqslant 100\,000$, $1 \leqslant m \leqslant 100\,000$) — число солдат и число приказов. Следующие m строк содержат приказы в виде двух целых чисел: l_i и r_i ($1 \leqslant l_i \leqslant r_i \leqslant n$).

Формат выходных данных

Выведите в выходной файл n целых чисел — порядок, в котором будут стоять солдаты после исполнения всех приказов.

Примеры

movetofront.in	movetofront.out
6 3 2 4 3 5 2 2	1 4 5 2 3 6

Задача C. Дерево

Имя входного файла:	`treenum.in`
Имя выходного файла:	`treenum.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Рассмотрим корневое дерево. Изначально дерево состоит из одного лишь корня. На сыновьях каждой вершины определён порядок слева направо. Допустимые операции с деревом таковы:

- Добавить в дерево лист.
- Удалить из дерева лист.
- Найти количество вершин на пути между двумя листьями.
- Найти количество вершин «под путём» между двумя листьями.

Множество вершин, лежащих «под путём» между листьями u и v, определяется следующим образом. Рассмотрим путь $u = w_0 - w_1 - \ldots - w_k = v$ между ними. Выделим в нём ту вершину w_c, в которой оба ребра пути идут в её сыновей. Пусть для определённости левое из этих рёбер приближает нас к вершине u, а правое — к вершине v. Тогда лежащими «под путём» объявляются следующие вершины:

- Все сыновья w_c, лежащие между w_{c-1} и w_{c+1}, а также все вершины их поддеревьев;
- Для $i = 1, 2, \ldots, c-1$ — все сыновья w_i, лежащие правее сына w_{i-1}, а также все вершины их поддеревьев;
- Для $i = c+1, c+2, \ldots, k-1$ — все сыновья w_i, лежащие левее сына w_{i+1}, а также все вершины их поддеревьев.

Напишите программу, которая по последовательности запросов перестраивает дерево согласно запросам на добавление и удаление вершин, а также вычисляет ответы на запросы о количестве вершин на пути и «под путём».

Формат входных данных

В первой строке ввода содержится одно целое число n — количество запросов ($0 \leqslant n \leqslant 300\,000$). Каждая из следующих n строк описывает один запрос. Возможные виды запросов таковы:

- `l` x — добавить новый лист как самого левого сына вершины x
- `r` x — добавить новый лист как самого правого сына вершины x
- `a` x y — добавить новый лист как сына вершины x, находящегося непосредственно справа от вершины y; все сыновья x, находившиеся до этого справа от y, после добавления оказываются правее новой вершины; гарантируется, что y является сыном x.
- `d` x удалить вершину x. Гарантируется, что в этот момент вершина x не удалена и является листом.
- `p` x y найти количество вершин на пути между x и y, включая сами эти вершины; гарантируется, что x и y являются листьями
- `q` x y найти количество вершин «под путём» между x и y, включая сами эти вершины; гарантируется, что x и y являются листьями.

Добавляемые вершины нумеруются с единицы в том порядке, в котором они добавляются в запросах. Корень дерева имеет номер 0 и листом ни в какой момент времени не считается.

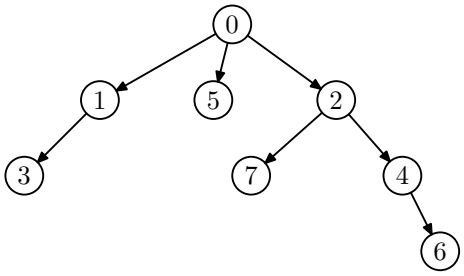
Формат выходных данных

Выполняя запросы по порядку, на каждый запрос вида `p` или `q` выведите ответ на него на отдельной строке.

Примеры

treenum.in	treenum.out
10 l 0 r 0 l 1 r 2 a 0 1 r 4 p 6 5 l 2 q 3 6 d 7	5 2

Замечание



На рисунке показано дерево до выполнения последнего запроса — удаления вершины 7.
Путь между вершинами 6 и 5 содержит пять вершин: 6 – 4 – 2 – 0 – 5.
«Под путём» между вершинами 3 и 6 находится две вершины — 5 и 7.
Эту задачу надо решать онлайн. Оффлайн решение получит Ignored.