Персистентный стек

Имя входного файла: `stack.in`
Имя выходного файла: `stack.out`

Реализуйте персистентный стек.

Формат входного файла

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

- `t m` — добавить в конец стека номер t ($0 \leqslant t < i$) число m ($0 < m \leqslant 1000$);
- `t 0` — удалить последний элемент стека номер t ($0 \leqslant t < i$). 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

Персистентная очередь

Имя входного файла: `queue.in`
Имя выходного файла: `queue.out`

Реализуйте персистентную очередь.

Формат входного файла

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

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

Откат

Имя входного файла: `rollback.in`
Имя выходного файла: `rollback.out`

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 ($1 \leqslant n, m \leqslant 100\,000$). Вторая строка содержит n целых чисел $a_1, a_2, \ldots, a_n$ — номера серверов, которым соответствуют точки восстановления ($1 \leqslant a_i \leqslant m$).

Третья строка входного файла содержит q — количество запросов, которые необходимо обработать ($1 \leqslant q \leqslant 100\,000$). 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: $l_i = ((x_i + p) \bmod n) + 1$, $k_i = ((y_i + p) \bmod m) + 1$ ($1 \leqslant l_i, x_i \leqslant n$, $1 \leqslant k_i, y_i \leqslant m$). 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