

Фабрика эхо

| | |
|-------------------------|-------------------|
| Имя входного файла: | стандартный ввод |
| Имя выходного файла: | стандартный вывод |
| Ограничение по времени: | 1 секунда |
| Ограничение по памяти: | 256 мегабайт |

Зельда нашла способ оптимизировать создание эхо объектов и создала конвейер по их производству.

Конвейер работает почти что по принципу обычной очереди, но также обладает и некоторыми дополнительными функциями: например, если где-то в очереди скопилось много эхо, которые Зельде уже не понадобятся, есть способ их быстро из очереди изъять.

Более подробно, конвейер в каждый момент времени может быть разделен на несколько сегментов, содержащих отрезки подряд идущих элементов очереди. Для работы с конвейером доступны следующие операции:

1. добавить элемент x в конец последнего сегмента очереди;
2. получить первый элемент первого сегмента очереди и удалить первый элемент **каждого** сегмента очереди; если какой-то сегмент после этого становится пустым, он удаляется, а его левый и правый соседи становятся соседними друг с другом;
3. разделить i -й по порядку сегмент очереди на первые j элементов и все оставшиеся; при такой операции i -й сегмент разбивается на два, а относительный порядок сегментов не меняется;
4. соединить все сегменты обратно в один сегмент, содержащий все элементы очереди.

Иными словами, порядок сохранившихся элементов в очереди не меняется, а при выполнении операции **pop** (2) помимо изъятия самого первого элемента очереди, также удаляются первые элементы каждого сегмента.

Для тестирования конвейера Зельда просит вас реализовать структуру данных, поддерживающую все эти операции.

Формат входных данных

В первой строке ввода даны целые числа n и q — изначальное количество элементов в очереди и число запросов ($1 \leq n, q \leq 10^5$).

Во второй строке ввода перечислены n целых чисел a_i — элементы очереди в ее начальном состоянии ($1 \leq a_i \leq 10^9$).

Далее следует q запросов, по одному в каждой строке. Формат запросов в соответствии с их описанием в условии следующий: «**push** x », «**pop**», «**cut** i j » и «**restore**» ($1 \leq x \leq 10^9$; $1 \leq i \leq S$, где S — количество непустых сегментов в данный момент; $1 \leq j < C_S$, где C_S — количество элементов в S -м сегменте).

Гарантируется, что все запросы корректны, в частности **pop** не вызывается при пустом первом сегменте, а **cut** не порождает пустые сегменты.

Формат выходных данных

На каждый запрос **pop** выведите на отдельной строке вынимаемый первый элемент очереди.

Гарантируется, что хотя бы один такой запрос будет.

Пример

| стандартный ввод | стандартный вывод |
|-----------------------|-------------------|
| 7 8 | 1 |
| 1 9 239 144 25 16 777 | 9 |
| pop | 239 |
| pop | 144 |
| cut 1 2 | 16 |
| pop | 777 |
| restore | |
| pop | |
| pop | |
| pop | |

Замечание

В первом примере из условия очередь претерпевает следующие изменения:

1. [1, 9, 239, 144, 25, 16, 777];
2. [9, 239, 144, 25, 16, 777];
3. [239, 144, 25, 16, 777];
4. [239, 144], [25, 16, 777];
5. [144], [16, 777];
6. [144, 16, 777];
7. [16, 777];
8. [777];
9. [];