

Использовать стандартную библиотеку (`priorityqueue`, `set`, `TreeSet`, и т. п.) не разрешается.

Задача А. Простое двоичное дерево поиска

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

Реализуйте двоичное дерево поиска.

Формат входного файла

Входной файл содержит описание операций с деревом, их количество не превышает 100. 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», если нет «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

В дерево помещаются и извлекаются только целые числа, не превышающие по модулю 10^9 .

Формат выходного файла

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

Пример

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

Задача В. Двоичное дерево поиска

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

Реализуйте двоичное дерево поиска. Вы должны реализовать именно то дерево, которое указано в вашем варианте.

Формат входного файла

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», если нет «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

В дерево помещаются и извлекаются только целые числа, не превышающие по модулю 10^9 .

Формат выходного файла

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

Пример

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

Задача С. Переместить в начало

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

Вам дан массив $a_1 = 1, a_2 = 2, \dots, a_n = n$ и последовательность операций: переместить элементы с l_i по r_i в начало массива. Например, для массива 2, 3, 6, 1, 5, 4, после операции (2, 4) новый порядок будет 3, 6, 1, 2, 5, 4. А после применения операции (3, 4) порядок элементов в массиве будет 1, 2, 3, 6, 5, 4.

Выведите порядок элементов в массиве после выполнения всех операций.

При реализации вы должны использовать то дерево поиска, которое указано в вашем варианте.

Формат входного файла

В первой строке входного файла указаны числа n и m ($2 \leq n \leq 100\,000$, $1 \leq m \leq 100\,000$) — число элементов в массиве и число операций. Следующие m строк содержат операции в виде двух целых чисел: l_i и r_i ($1 \leq l_i \leq r_i \leq n$).

Формат выходного файла

Выведите n целых чисел — порядок элементов в массиве после применения всех операций.

Пример

<code>movetofront.in</code>	<code>movetofront.out</code>
6 3 2 4 3 5 2 2	1 4 5 2 3 6