

## Задача А. Прибавление и максимум

Имя входного файла: `addandmax.in`  
Имя выходного файла: `addandmax.out`  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 64 мегабайта

Реализуйте эффективную структуру данных для хранения массива и выполнения следующих операций: увеличение всех элементов данного интервала на одно и то же число; поиск максимума на интервале.

### Формат входных данных

В первой строке вводится одно натуральное число  $N$  ( $1 \leq N \leq 100000$ ) – количество чисел в массиве.

Во второй строке вводятся  $N$  чисел от 0 до 100000 – элементы массива.

В третьей строке вводится одно натуральное число  $M$  ( $1 \leq M \leq 30000$ ) – количество запросов.

Каждая из следующих  $M$  строк представляет собой описание запроса. Сначала вводится одна буква, кодирующая вид запроса ( $m$  – найти максимум,  $a$  – увеличить все элементы на отрезке).

Следом за  $m$  вводятся два числа – левая и правая граница интервала.

Следом за  $a$  вводятся три числа – левый и правый концы отрезка и число  $add$ , на которое нужно увеличить все элементы данного отрезка массива ( $0 \leq add \leq 100000$ ).

### Формат выходных данных

Выведите в одну строку через пробел ответы на каждый запрос  $m$ .

### Примеры

<code>addandmax.in</code>	<code>addandmax.out</code>
5	4
2 4 3 1 5	104
5	104
m 1 3	
a 2 4 100	
m 1 3	
a 5 5 10	
m 1 5	

## Задача В. Декартово дерево

Имя входного файла: `tree.in`  
Имя выходного файла: `tree.out`  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 256 мегабайт

Вам даны пары чисел  $(a_i, b_i)$ . Необходимо построить декартово дерево, такое что  $i$ -я вершина имеет ключи  $(a_i, b_i)$ , вершины с ключом  $a_i$  образуют бинарное дерево поиска, а вершины с ключом  $b_i$  образуют кучу.

### Формат входных данных

В первой строке записано число  $N$  — количество пар. Далее следует  $N$  ( $1 \leq N \leq 50\,000$ ) пар  $(a_i, b_i)$ . Для всех пар  $|a_i|, |b_i| \leq 30\,000$ .  $a_i \neq a_j$  и  $b_i \neq b_j$  для всех  $i \neq j$ .

### Формат выходных данных

Если декартово дерево с таким набором ключей построить возможно, выведите в первой строке «YES», в противном случае выведите «NO». В случае ответа «YES» выведите  $N$  строк, каждая из которых должна описывать вершину. Описание вершины состоит из трёх чисел: номера предка, номера левого сына и номера правого сына. Если у вершины отсутствует предок или какой либо из сыновей, выведите на его месте число 0.

Если подходящих деревьев несколько, выведите любое.

### Пример

tree.in	tree.out
7	YES
5 4	2 3 6
2 2	0 5 1
3 9	1 0 7
0 5	5 0 0
1 3	2 4 0
6 6	1 0 0
4 11	3 0 0

## Задача С. Двоичное дерево поиска 1

Имя входного файла: `bst1.in`  
Имя выходного файла: `bst1.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Реализуйте сбалансированное двоичное дерево поиска. Внимание! Решать задачу с использованием `set` из STL запрещено, однако рекомендуется стрессить ваше решение с ним для поиска багов

### 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  $x$ . Если ключ  $x$  уже в дереве, то ничего делать не надо.
- `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»`

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

### Формат выходных данных

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

### Примеры

<code>bst1.in</code>	<code>bst1.out</code>
<code>insert 2</code>	<code>true</code>
<code>insert 5</code>	<code>false</code>
<code>insert 3</code>	
<code>exists 2</code>	
<code>exists 4</code>	
<code>delete 5</code>	

## Задача D. Двоичное дерево поиска 2

Имя входного файла: `bst2.in`  
Имя выходного файла: `bst2.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. Формат операций смотрите в предыдущей задаче. 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.
- `kth k` — выведите  $k$ -ый по величине элемент (нумерация с единицы). Если такого не существует, то выведите «`none`».

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

bst2.in	bst2.out
<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>2</code>
<code>delete 5</code>	<code>none</code>
<code>next 4</code>	
<code>prev 4</code>	
<code>kth 1</code>	
<code>kth 3</code>	