Задача 1.

Средняя

**Декартово дерево - 1**

**Входной файл: input.txt или стандартный ввод**

**Выходной файл: output.txt или стандартный вывод**

**Ограничение по времени работы: 1 секунда**

**В этой задаче нельзя использовать стандартные контейнеры set и map (нельзя подключать заголовочные файлы set, map, а также bits/stdc++.h).**

Реализуйте сбалансированное двоичное дерево поиска для хранения целых чисел (значений типа int).

ВХОДНЫЕ ДАННЫЕ

Первая строка входных данных содержит количество операций с деревом N (N⩽100000).

Следующие N строк содержат описание операций с деревом. В каждой строке содержится одна из следующих операций:

1. **insert** x — добавить в дерево ключ x. Если ключ x уже в дереве, то ничего делать не надо.
2. **delete** x — удалить из дерева ключ x. Если ключа x нет в дереве, то ничего делать не надо.
3. **exists** x —  если ключ x есть в дереве, то нужно вывести **true**, иначе вывести **false**.

ВЫХОДНЫЕ ДАННЫЕ

Для каждого запроса **exists** программа должна вывести **true** или **false** в отдельной строке.

ПРИМЕР

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 6 insert 2 insert 5 insert 3 exists 2 exists 4 delete 5 | true false |

Задача 2.

Сложная

**Декартово дерево - 2**

**Ограничение по времени работы: 1 секунда**

Реализуйте сбалансированное двоичное дерево поиска, которое умеет не только проверять наличие элемента в дереве, но и определять, каким по счёту является этот элемент в дереве (считая, что элементы упорядочены по возрастанию ключей).

ВХОДНЫЕ ДАННЫЕ

Первая строка входных данных содержит количество операций с деревом N (N⩽100000).

Следующие N строк содержат описание операций с деревом. В каждой строке содержится одна из следующих операций:

1. **insert** x — добавить в дерево ключ x. Если ключ x уже в дереве, то ничего делать не надо.
2. **delete** x — удалить из дерева ключ x. Если ключа x нет в дереве, то ничего делать не надо.
3. **exists** x — если ключ x есть в дереве, то нужно вывести его порядковый номер в дереве (1 — если ключ x является минимальным элементом в дереве, 2 — если он является следующим за минимальным и т.д.). Если этого элемента нет в дереве, нужно вывести 0.

ВЫХОДНЫЕ ДАННЫЕ

Для каждого запроса **exists** программа должна вывести одно целое неотрицательное число — ответ на запрос.

ПРИМЕР

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 6 insert 2 insert 5 insert 3 exists 2 exists 4 delete 5 | 1 0 |

Задача 3.

Сложная

**Строй**

**Ограничение по времени работы: 1 секунда**

В одной военной части решили построить в одну шеренгу по росту. Т.к. часть была далеко не образцовая, то солдаты часто приходили не вовремя, а то их и вовсе приходилось выгонять из шеренги за плохо начищенные сапоги. Однако солдаты в процессе прихода и ухода должны были всегда быть выстроены по росту – сначала самые высокие, а в конце – самые низкие. За расстановку солдат отвечал прапорщик, который заметил интересную особенность – все солдаты в части разного роста. Ваша задача состоит в том, чтобы помочь прапорщику правильно расставлять солдат, а именно для каждого приходящего солдата указывать, перед каким солдатом в строе он должен становится.

### ВХОДНЫЕ ДАННЫЕ

Первая строка содержит число N — количество команд (1⩽N⩽100000). В каждой следующей строке содержится описание команды: числа 1 и X, если солдат приходит в строй (X — рост солдата, натуральное число до 100000 включительно) и число 2 и Y если солдата, стоящим в строе на месте Y надо удалить из строя.

### ВЫХОДНЫЕ ДАННЫЕ

На каждую команду 1 (добавление в строй) вы должны выводить число K — номер позиции, на которую должен встать этот солдат (все стоящие за ним двигаются назад). Выводите числа по одному в строке.

### ПРИМЕР

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 5  1 100  1 200  1 50  2 2  1 150 | 1  1  3  2 |

Задача 4.

Сложная

**Построение декартова дерева**

**Ограничение по времени работы: 1 секунда**

Даны пары чисел (ai,bi). Необходимо построить декартово дерево так, чтобы в вершине номер i были ключи (ai,bi), вершины с ключом ai образуют двоичное дерево поиска (слева меньшие значения, справа — большие), а значения bi образуют кучу (в вершине — наименьший элемент).

### ВХОДНЫЕ ДАННЫЕ

В первой строке входных данных записано число N (1⩽N⩽50000) — количество пар. Следующие N строк содержат пары чисел (ai,bi). Для всех чисел |ai|<30000, |bi|<30000, ai≠aj и bi≠bj для всех i≠j.

### ВЫХОДНЫЕ ДАННЫЕ

Программа должна вывести N строк, каждая из которых описывает вершины декартова дерева в том порядке, в котором они следуют во входном файле (вершины пронумерованы числами от 1 до N в порядке из следования в файле). Описание вершины состоит из трех чисел — номера предка этой вершины, номера левого потомка, номера правого потомка. Если у вершины отсутствует предок или какой-либо из потомков, вместо его номера нужно вывести число 0.

### ПРИМЕР

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 7 5 4 2 2 3 9 0 5 1 3 6 6 4 11 | 2 3 6 0 5 1 1 0 7 5 0 0 2 4 0 1 0 0 3 0 0 |

Задача 5.

Сложная

**И снова сумма...**

Реализуйте структуру данных, которая поддерживает  
множество S целых чисел, с котором разрешается производить  
следующие операции:  
add(i) -— добавить в множество S число i (если он  
там уже есть, то множество не меняется);  
sum(l,r) -— вывести сумму всех элементов x из S, которые  
удовлетворяют неравенству l≤x≤r.  
\end{itemize}

### ВХОДНЫЕ ДАННЫЕ

Исходно множество S пусто. Первая строка входного файла содержит  
n -— количество операций (1≤n≤300000).  
Следующие n строк содержат операции.  
Каждая операция имеет вид либо +  i, либо ?  l  r.  
Операция ?  l  r. задает запрос sum(l,r). Если операция +  i идет во входном файле в начале или  
после другой операции +, то она задает операцию add(i).  
Если же она идет после запроса ?, и результат  
этого запроса был y, то выполняется операция add((i+y)mod109). Во всех запросах и операциях добавления параметры лежат в интервале от 0 до 109.

### ВЫХОДНЫЕ ДАННЫЕ

Для каждого запроса выведите одно число -— ответ на запрос.

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 6 + 1 + 3 + 3 ? 2 4 + 1 ? 2 4 | 3 7 |

Задача 6.

Олимпиадная

**Декартово дерево-3**

Реализуйте сбалансированное двоичное дерево поиска.

### ВХОДНЫЕ ДАННЫЕ

Входной файл содержит описание операций с деревом. Операций не больше 105. 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, true, иначе false  
next x — минимальный элемент в дереве, >x, или none, если такого нет.  
prev x — максимальный элемент в дереве, <x, или none, если такого нет.  
  
Все числа во входном файле целые и по модулю не превышают 109.

### ВЫХОДНЫЕ ДАННЫЕ

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

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 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 |