Задача A. 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

bst1.in	bst1.out
insert 2 insert 5 insert 3 exists 2 exists 4 delete 5	true false

Задача B. 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`, `kth`. Следуйте формату выходного файла из примера.

Примеры

bst2.in	bst2.out
insert 2 insert 5 insert 3 exists 2 exists 4 next 4 prev 4 delete 5 next 4 prev 4 kth 1 kth 3	true false 5 3 none 3 2 none

Задача C. И снова сумма...

Имя входного файла:	`sum2.in`
Имя выходного файла:	`sum2.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

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 $l \leqslant x \leqslant r$.

Формат входных данных

Исходно множество S пусто. Первая строка входного файла содержит n — количество операций ($1 \leqslant n \leqslant 300\,000$). 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 $add((i + y) \bmod 10^9)$.

Во всех запросах и операциях добавления параметры лежат в интервале от 0 до 10^9.

Формат выходных данных

Для каждого запроса выведите одно число — ответ на запрос.

Примеры

sum2.in	sum2.out
6 + 1 + 3 + 3 ? 2 4 + 1 ? 2 4	3 7

Задача D. 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 ($1 \leqslant N \leqslant 50\,000$) пар (a_i, b_i). Для всех пар $|a_i|, |b_i| \leqslant 30\,000$. $a_i \neq a_j$ 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 5 4 2 2 3 9 0 5 1 3 6 6 4 11	YES 2 3 6 0 5 1 1 0 7 5 0 0 2 4 0 1 0 0 3 0 0

Задача E. Легчайшая

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

Имеется N наборов чисел, которые изначально известны.

Поступают Q запросов вида x y k, для выполнения запроса нужно взять все числа из набора под номером x, которые не меньше числа k, и переместить их в набор номер y. После выполнения всех запросов необходимо вывести конечные состояния наборов.

Формат входных данных

В первой строке дано два числа — N и Q ($2 \leqslant N \leqslant 100,000$, $1 \leqslant Q \leqslant 100,000$).

В последующих N строках заданы наборы. Каждый набор задается строкой вида: число k, за ним k чисел в неубывающем порядке. Суммарный размер наборов не превышает 100,000. Все числа в наборах от 1 до 1,000,000.

Далее в Q строках заданы запросы — по три числа x, y и k ($1 \leqslant x, y, \leqslant N$, $x \neq y$, $1 \leqslant k \leqslant 1,000,000$).

Формат выходных данных

Выведите N строк — конечные состояния наборов, выводить числа набора следует в неубывающем порядке.

Примеры

theeasiest.in	theeasiest.out
3 2 3 1 2 3 3 1 2 4 0 1 2 2 2 3 3	1 1 3 1 2 2 2 3 4