Содержание

Обратите внимание, входные данные лежат в **стандартном потоке ввода** (он же stdin), вывести ответ нужно в **стандартный поток вывода** (он же stdout).

Обратите внимание на `GNU C++` компиляторы с суффиксом `inc`.
Подни можно пользоваться дополнительной библиотекой (`optimization.h`).
То есть, использовать быстрый ввод-вывод: **пример про числа и строки**.
И быструю аллокацию памяти (ускоряет vector-set-map-весь-STL): пример.

Для тех, кто хочет разобраться, как всё это работает.
Короткая версия быстрого ввода-вывода (тык) и короткая версия аллокатора (тык).

Must have

Задача 17A. Простейшее BST [0.2, 256]

В этой задаче вам нужно написать простейшее BST по явному ключу и отвечать им на запросы:

- `+ x` – добавить в дерево `x` (если `x` уже есть, ничего не делать).
- `> x` – вернуть минимальный элемент больше `x` или 0, если таких нет.

Формат входных данных

В каждой строке содержится один запрос.
Все x целые от 1 до 10^9, количество запросов от 1 до 300 000.
Гарантируется, что все x выбраны равномерным распределением.

Формат выходных данных

Для каждого запроса вида «`> x`» выведите в отдельной строке ответ.

Пример

stdin	stdout
+ 1 + 3 + 3 > 1 > 2 > 3 + 2 > 1	3 3 0 2

Задача 17B. Простейший полуявный ключ [0.2, 256]

В этой задаче вам нужно написать BST по **явному** ключу и отвечать им на запросы:

- `+ x` – добавить в дерево `x` (если `x` уже есть, ничего не делать).
- `? k` – вернуть k-й по возрастанию элемент.

Формат входных данных

В каждой строке содержится один запрос.
Все x целые от 1 до 10^9, количество запросов от 1 до 300 000.
Гарантируется, что все x выбраны равномерным распределением.
В запросах «`? k`», число k от 1 до количества элементов в дереве.

Формат выходных данных

Для каждого запроса вида «`? k`» выведите в отдельной строке ответ.

Пример

stdin	stdout
+ 1 + 4 + 3 + 3 ? 1 ? 2 ? 3 + 2 ? 3	1 3 4 3

Подсказка по решению

Случайные данные! Не нужно ничего специально балансировать.

Вам нужно в каждой вершине поддерживать размер поддерева.
В этой задаче вам нужно написать BST по **не**явному ключу.

Задача 17C. Простейший неявный Ключ [0.2, 256]

Изначально есть пустой массив. Вам нужно обрабатывать запросы вида
`i x` — добавить после i-го элемента x $(0 \leqslant i \leqslant n)$, где n текущая длина массива.
В конце после всех добавлений нужно вывести полученный массив.

Формат входных данных

q $(1 \leqslant q \leqslant 100\,000)$ строк, на каждой запрос «`i x`» —
пара целых чисел $(0 \leqslant i \leqslant n,\ 1 \leqslant x < 100\,000)$.

Формат выходных данных

Выведите q целых чисел — полученный массив.

Примеры

stdin	stdout
0 1 0 2 0 3	3 2 1
0 1 1 2 2 3	1 2 3
0 1 1 2 1 3 0 4	4 1 3 2

Подсказка по решению

Случайные данные! Не нужно ничего специально балансировать.

Вам нужно в каждой вершине поддерживать размер поддерева.
В этой задаче вам нужно написать BST по **не**явному ключу.

Задачи здорового человека

Задача 17D. Дебаг-вывод дерева [0.2, 256]

Дана последовательность добавлений в дерево поиска (bst). После каждого добавления выводите LxR обход дерева (вывести левое поддерева, вывести ключ, вывести правое поддерева).

Формат входных данных

Последовательность из n ($1 \leqslant n \leqslant 1\,000$) целых чисел от 0 до 10^6, которые следуют добавить. Добавлять нужно именно в таком порядке. Гарантируется, что все числа **различны**.

Формат выходных данных

После каждого добавления выведите дерево в особом формате (см. примеры). Деревья для удобства чтения разделяйте строкой с одним дефисом. Обратите внимание, число пробелов перед x равно 2· глубина вершины.

Примеры

stdin	stdout

1
5
9
3
	x=1 size=1
	-
	x=1 size=2
	  x=5 size=1
	-
	x=1 size=3
	  x=5 size=2
	  x=9 size=1
	-
	x=1 size=4
	  x=3 size=1
	  x=5 size=3
	  x=9 size=1

Задача 17E. Двоичное дерево поиска [0.1, 256]

Реализуйте сбалансированное двоичное дерево поиска.
Или воспользуйтесь любой стандартной структурой из `C++::STL`.
Или попытайтесь запихать квадрат.

Формат входных данных

Входной файл содержит описание одной или нескольких операций с деревом.
Операций не больше 10^5. Все числа целые от -10^5 до 10^9.
В каждой строке находится одна из следующих операций:

- `insert` x — добавить в дерево ключ x.
  Если ключ x в дереве уже есть, 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, «`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.

Формат выходных данных

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

Примеры

stdin	stdout
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

Подсказка по решению

`while (!seekEof()) { ... }`
Заходят `set`, умный квадрат, сбалансированное bst.

Задача 17F. K-ый максимум [0.1, 256]

Напишите программу, реализующую структуру данных, позволяющую добавлять и удалять элементы, а также находить k-й максимум.

Формат входных данных

Первая строка входного файла содержит натуральное число n — количество команд ($n \leqslant 100\,000$). Последующие n строк содержат по одной команде каждая. Команда записывается в виде двух чисел c_i и k_i — тип и аргумент команды соответственно ($|k_i| \leqslant 10^9$). Поддерживаемые команды:

- $+1$ (или просто 1): Добавить элемент с ключом k_i.
- 0: Найти и вывести k_i-й максимум.
- -1: Удалить элемент с ключом k_i.

Гарантируется, что в процессе работы в структуре не требуется хранить элементы с равными ключами или удалять несуществующие элементы. Также гарантируется, что при запросе k_i-го максимума, он существует.

Формат выходных данных

Для каждой команды нулевого типа в выходной файл должна быть выведена строка, содержащая единственное число — k_i-й максимум.

Пример

stdin	stdout
11 +1 5 +1 3 +1 7 0 1 0 2 0 3 -1 5 +1 10 0 1 0 2 0 3	7 5 3 10 7 3

Для искателей острых ощущений

Задача 17G. И снова сумма... [0.5, 256]

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.

Пример

stdin	stdout
6 + 1 + 3 + 3 ? 2 4 + 1 ? 2 4	3 7