

Задача А. Мультисет

Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Обработайте запросы двух видов:

"*add x*" — добавить число x в мультимножество.

"*ask x*" — вывести количество вхождений числа x в мультимножество.

Формат входных данных

В первой строке вводится одно число n ($1 \leq n \leq 300\,000$) — число запросов. В каждой из следующих n строк вводятся строка s и число x ($1 \leq x \leq 10^9$), описывающие запрос. Гарантируется, что есть хотя бы один запрос типа "*ask*".

Формат выходных данных

На каждый запрос типа "*ask*" надо вывести в отдельной строке количество вхождений числа x в мультимножество.

Примеры

стандартный ввод	стандартный вывод
4	0
ask 4	1
add 4	0
ask 4	
ask 5	

Задача В. Двоичное дерево поиска

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

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

<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. 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 которым 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 $l \leq x \leq r$.

Формат входных данных

Исходно множество S пусто. Первая строка входного файла содержит n — количество операций ($1 \leq n \leq 300\,000$). 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

<code>sum2.in</code>	<code>sum2.out</code>
6	3
+ 1	7
+ 3	
+ 3	
? 2 4	
+ 1	
? 2 4	

Задача D. К-ый максимум

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

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 ($n \leq 100\,000$). Последующие n строк содержат по одной команде каждая. Команда записывается в виде двух чисел c_i и k_i — тип и аргумент команды соответственно ($|k_i| \leq 10^9$). 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.

Примеры

<code>kthmax.in</code>	<code>kthmax.out</code>
11	7
+1 5	5
+1 3	3
+1 7	10
0 1	7
0 2	3
0 3	
-1 5	
+1 10	
0 1	
0 2	
0 3	