

Задача А. Следующий

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Реализуйте структуру данных, которая поддерживает множество S целых чисел, с которым разрешается производить следующие операции:

1. `add(i)` – добавить в множество S число i (если он там уже есть, то множество не меняется)
2. `next(i)` – вывести минимальный элемент множества, не меньший i . Если искомый элемент в структуре отсутствует, необходимо вывести -1 .

Заметьте, что в этой задаче необычные операции ввода. Операция, которую вам задает тест, может зависеть от того, правильно ли вы ответили на предыдущий запрос. Внимательно прочитайте формат ввода. Операция *mod* означает взятие остатка.

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 `«? i»`. Операция `«? i»` задает запрос `next(i)`.

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) mod 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

стандартный ввод	стандартный вывод
6	3
+ 1	4
+ 3	
+ 3	
? 2	
+ 1	
? 4	

Задача В. Два-Три-Де... Дерево

Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 2.3 секунд
Ограничение по памяти: 256 мегабайт

В этой задаче от вас требуется реализовать добавление элемента в структуру данных «2-3 дерево». Для того, чтобы построение дерева было однозначным, корректно определим операцию добавления листа. А именно, если мы добавляем значение x , и находимся в вершине v , у которой есть два соседних сына u_1 и u_2 , то тогда, если $\max(subtree(u_1)) < x < \min(subtree(u_2))$ (то есть, новый лист может быть самым правым в левом поддереве или самым левым в правом), мы добавляем лист в правое поддерево.

После того, как вы построите 2-3 дерево, сделайте рекурсивную процедуру обхода, чтобы вывести его.

Формат входных данных

В первой строке вводится натуральное число n ($1 \leq n \leq 10^6$) — количество запросов на добавление в 2-3 дерево. Во второй строке вводится n различных натуральных чисел a_i ($1 \leq a_i \leq n$) — запросы добавления. Обратите внимание, что выполнять запросы нужно именно в этом порядке.

Формат выходных данных

Выведите все листья дерева в отсортированном порядке. Разделяйте два соседних числоа латинской буквой, которая будет соответствовать глубине LCA этих вершин. Будем считать, что корню соответствует символ 'А', вершинам на глубине 1 соответствует символ 'В', и так далее. В вашем выводе должно оказаться n чисел и $n - 1$ латинский символ.

Примеры

стандартный ввод	стандартный вывод
10 1 2 3 4 5 6 7 8 9 10	1 С 2 В 3 С 4 А 5 С 6 В 7 С 8 В 9 С 10
2 1 2	1 А 2

Задача С. К-ый максимум

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	64 мегабайта

Напишите программу, реализующую структуру данных, позволяющую добавлять и удалять элементы, а также находить k -й максимум. Вам достаточно поддерживать в вершине количество элементов в ее поддереве, обновлять эту величину при каждом обновлении вершины, а затем делать спуск.

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.

Пример

стандартный ввод	стандартный вывод
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	

Задача D. И снова сумма...

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
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 $l \leq x \leq r$.

Обратите внимание, что ответ на запрос зависит от ответа на предыдущий запрос

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.

Пример

стандартный ввод	стандартный вывод
6	3
+ 1	7
+ 3	
+ 3	
? 2 4	
+ 1	
? 2 4	