Задача A. Вставка ключевых значений

Имя входного файла:	`key.in`
Имя выходного файла:	`key.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Вас наняла на работу компания MacroHard, чтобы вы разработали новую структуру данных для хранения целых ключевых значений.

Эта структура выглядит как массив A бесконечной длины, ячейки которого нумеруются с единицы. Изначально все ячейки пусты. Единственная операция, которую необходимо поддерживать — это операция $Insert(L, K)$, где L — положение в массиве, а K — некоторое положительное целое ключевое значение.

Операция выполняется следующим образом:

- Если ячейка $A[L]$ пуста, то присвоить $A[L] := K$.
- Если ячейка $A[L]$ непуста, выполнить $Insert(L + 1, A[L])$, а затем присвоить $A[L] := K$.

По заданной последовательности из N целых чисел $L_1, L_2, \ldots, L_N$ вам необходимо вывести содержимое этого массива после выполнения следующей последовательности операций:

$Insert(L_1, 1)$
$Insert(L_2, 2)$
...
$Insert(L_N, N)$

Формат входного файла

В первой строке входного файла содержится N — число операций $Insert$ и M — максимальный номер позиции, которую можно использовать в операции $Insert$. ($1 \le N \le 131\,072$, $1 \le M \le 131\,072$).

В следующей строке даны N целых чисел L_i, которые описывают операции $Insert$ ($1 \le L_i \le M$).

Формат выходного файла

Выведите содержимое массива после выполнения данной последовательности операций $Insert$. На первой строке выведите W — номер последней несвободной позиции в массиве. Далее выведите W целых чисел — $A[1], A[2], \ldots, A[W]$. Для пустых ячеек выводите нули.

Пример

key.in	key.out
5 4 3 3 4 1 3	6 4 0 5 2 3 1

Задача B. Вставка ключевых значений-easy

Задача «Вставка ключевых значений» с ограничениями $1 \le N \le 300$, $1 \le M \le 300$.

Задача C. Вперёд!

Имя входного файла:	`movetofront.in`
Имя выходного файла:	`movetofront.out`
Ограничение по времени:	6 секунд
Ограничение по памяти:	256 мегабайт

Капрал Дукар любит раздавать приказы своей роте. Самый любимый его приказ — "Вперёд!". Капрал строит солдат в ряд и отдаёт некоторое количество приказов, каждый из них звучит так: "Рядовые с l_i по l_j — вперёд!"

Перед тем, как Дукар отдал первый приказ, солдаты были пронумерованы от 1 до n, слева направо. Услышав приказ "Рядовые с l_i по l_j — вперёд!", солдаты, стоящие на местах с l_i по l_j включительно, продвигаются в начало ряда, в том же порядке, в котором были.

Например, если в какой-то момент солдаты стоят в порядке $1, 3, 6, 2, 5, 4$, то после приказа "Рядовые с 2 по 3 — вперёд!", порядок будет таким: $3, 6, 1, 2, 5, 4$. А если потом Капрал вышлет вперёд солдат с 3 по 4, то порядок будет уже таким: $1, 2, 3, 6, 5, 4$.

Вам дана последовательность из приказов Капрала. Найдите порядок, в котором будут стоять солдаты после исполнения всех приказов.

Формат входного файла

В первой строке входного файла указаны числа n и m ($2 \le n \le 100\,000$, $1 \le m \le 100\,000$) — число солдат и число приказов. Следующие m строк содержат приказы в виде двух целых чисел: l_i и r_i ($1 \le l_i \le r_i \le n$).

Формат выходного файла

Выведите в выходной файл n целых чисел — порядок, в котором будут стоять солдаты после исполнения всех приказов.

Пример

movetofront.in	movetofront.out
6 3 2 4 3 5 2 2	1 4 5 2 3 6

Задача D. Вперёд!-easy

Задача «Вперёд!» с ограничениями $1 \le n, m \le 100$.

Задача E. И снова сумма...

Имя входного файла:	`sum.in`
Имя выходного файла:	`sum.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	64 мегабайта

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 $l \le x \le r$.

Формат входного файла

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

Для каждого запроса выведите одно число — ответ на запрос.

Пример

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

Задача F. И снова сумма... (easy)

Задача «И снова сумма...» с ограничениями $1 \le N \le 100$

Задача G. Обмен

Имя входного файла:	`swap.in`
Имя выходного файла:	`swap.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Пусть все натуральные числа исходно организованы в список в естественном порядке. Разрешается выполнить следующую операцию: $swap(a, b)$. Эта операция возвращает в качестве результата расстояние в текущем списке между числами a и b и меняет их местами.

Задана последовательность операций *swap*. Требуется вывести в выходной файл результат всех этих операций.

Формат входного файла

Первая строка входного файла содержит число n ($1 \le n \le 200\,000$) — количество операций. Каждая из следующих n строк содержит по два числа в диапазоне от 1 до 10^9 — аргументы операций *swap*.

Формат выходного файла

Для каждой операции во входном файле выведите ее результат.

Пример

swap.in	swap.out
4 1 4 1 3 4 5 1 4	3 1 4 2