

Problem A. Снеговики

Input file: snowmen.in
Output file: snowmen.out
Time limit: 4 секунды
Memory limit: 64 мегабайта

Зима. 2012 год. На фоне грядущего Апокалипсиса и конца света незамеченной прошла новость об очередном прорыве в областях клонирования и снеговиков: клонирования снеговиков. Вы конечно знаете, но мы вам напомним, что снеговик состоит из нуля или более вертикально поставленных друг на друга шаров, а клонирование — это процесс создания идентичной копии (клона).

В местечке Местячково учитель Андрей Сергеевич Учитель купил через интернет-магазин «Интернет-магазин аппаратов клонирования» аппарат для клонирования снеговиков. Теперь дети могут играть и даже играют во дворе в следующую игру. Время от времени один из них выбирает понравившегося снеговика, клонирует его и:

- либо добавляет ему сверху один шар;
- либо удаляет из него верхний шар (если снеговик не пустой).

Учитель Андрей Сергеевич записал последовательность действий и теперь хочет узнать суммарную массу всех построенных снеговиков.

Input

Первая строка содержит количество действий n ($1 \leq n \leq 200\,000$). В строке номер $i + 1$ содержится описание действия i :

- $t\ m$ — клонировать снеговика номер t ($0 \leq t < i$) и добавить сверху шар массой m ($0 < m \leq 1000$);
- $t\ 0$ — клонировать снеговика номер t ($0 \leq t < i$) и удалить верхний шар. Гарантируется, что снеговик t не пустой.

В результате действия i , описанного в строке $i + 1$ создается снеговик номер i . Изначально имеется пустой снеговик с номером ноль.

Все числа во входном файле целые.

Output

Выведите суммарную массу построенных снеговиков.

Examples

snowmen.in	snowmen.out
8	74
0 1	
1 5	
2 4	
3 2	
4 3	
5 0	
6 6	
1 0	

Problem B. НВП на дереве

Input file: стандартный ввод
Output file: стандартный вывод
Time limit: 2 секунды
Memory limit: 256 мегабайт

Данную задачу необходимо сдать частичной персистентностью(метод откатов). Дано дерево на N вершинах, у которого i -е ребро соединяет вершину u_i и вершину v_i . На вершине с номером i записано целое число a_i . Для каждого целого числа k от 1 до N решите следующую задачу:

Составим последовательность, выписав целые числа на вершинах, вдоль кратчайшего пути от вершины 1 до вершины k , в том порядке, в котором они появляются. Найдите длину наибольшей возрастающей подпоследовательности этой последовательности.

Input

В первой строке вводится число n ($2 \leq n \leq 2 \cdot 10^5$) — количество вершин в дереве.

Во второй строке через пробел задаются числа a_i ($1 \leq a_i \leq 10^9$) — числа, записанные на вершинах.

В каждой из следующих $n - 1$ -й строке вводятся пары v_i, u_i — рёбра дерева ($1 \leq v_i \neq u_i \leq n$). Гарантируется, чтоданный набор рёбер образует дерево.

Output

Выведите N строк. В k -й строке выведите длину наибольшей возрастающей подпоследовательности последовательности, полученной по кратчайшему пути из вершины 1 в вершину k .

Examples

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

Problem C. Persistent Array

Input file: стандартный ввод
Output file: стандартный вывод
Time limit: 2 секунды
Memory limit: 256 мегабайт

Дан массив (вернее, первая, начальная его версия).

Нужно уметь отвечать на два запроса:

- $a_i[j] = x$ — создать из i -й версии новую, в которой j -й элемент равен x , а остальные элементы такие же, как в i -й версии.
- `get $a_i[j]$` — сказать, чему равен j -й элемент в i -й версии.

Input

Количество чисел в массиве N ($1 \leq N \leq 10^5$) и N элементов массива. Далее количество запросов M ($1 \leq M \leq 10^5$) и M запросов. Формат описания запросов можно посмотреть в примере. Если уже существует K версий, новая версия получает номер $K + 1$. И исходные, и новые элементы массива — целые числа от 0 до 10^9 . Элементы в массиве нумеруются числами от 1 до N .

Output

На каждый запрос типа `get` вывести соответствующий элемент нужного массива.

Examples

стандартный ввод	стандартный вывод
6	6
1 2 3 4 5 6	5
11	10
create 1 6 10	5
create 2 5 8	10
create 1 5 30	8
get 1 6	6
get 1 5	30
get 2 6	
get 2 5	
get 3 6	
get 3 5	
get 4 6	
get 4 5	

Problem D. Различные числа online

Input file: стандартный ввод
Output file: стандартный вывод
Time limit: 7 секунд
Memory limit: 512 мегабайт

Сколько различных чисел на отрезке массива?

Input

На первой строке длина массива n ($1 \leq n \leq 300\,000$). На второй строке n целых чисел от 0 до 10^9-1 . На третьей строке количество запросов q ($1 \leq q \leq 300\,000$). Следующие q строк содержат описание запросов, по одному на строке. Каждый запрос задаётся парой целых чисел l, r ($1 \leq l \leq r \leq n$).

Output

Выведите ответы на запросы по одному в строке.

Examples

стандартный ввод	стандартный вывод
5	3
1 1 2 1 3	2
3	3
1 5	
2 4	
3 5	

Note

В этой задаче необходимо отвечать на запросы **online**, поэтому задача настроена как интерактивная. После вывода ответа на запрос, вы должны сбросить буфер вывода до начала ввода следующего запроса.

Problem E. CHM

Input file: стандартный ввод
Output file: стандартный вывод
Time limit: 2 секунды
Memory limit: 256 мегабайт

Ваша задача — реализовать **Persistent Disjoint-Set-Union**. Что это значит?

Про **Disjoint-Set-Union**:

Изначально у вас есть n элементов. Нужно научиться отвечать на 2 типа запросов.

- $+ a b$ — объединить множества, в которых лежат вершины a и b
- $? a b$ — сказать, лежат ли вершины a и b сейчас в одном множестве

Про **Persistent**:

Теперь у нас будет несколько копий (версий) структуры данных **Disjoint-Set-Union**.

Запросы будут выглядеть так:

- $+ i a b$ — запрос к i -й структуре, объединить множества, в которых лежат вершины a и b . При этом i -я структура остается не измененной, создается новая версия, ей присваивается новый номер (какой? читайте дальше)
- $? i a b$ — запрос к i -й структуре, сказать, лежат ли вершины a и b сейчас в одном множестве

Input

На первой строке 2 числа N ($1 \leq N \leq 10^5$) и K ($0 \leq K \leq 10^5$) — число элементов и число запросов. Изначально все элементы находятся в различных множествах. Эта начальная копия (версия) структуры имеет номер 0.

Далее следуют K строк, на каждой описание очередного запроса. Формат запросов описан выше. Запросы нумеруются целыми числами от 1 до K .

Пусть j -й из K запросов имеет вид « $+ i a b$ ». Тогда новая версия получит номер j . Запросы вида « $? i a b$ » не порождают новых структур.

Output

Для каждого запроса вида $? i a b$ на отдельной строке нужно вывести YES или NO.

Examples

стандартный ввод	стандартный вывод
4 7	NO
+ 0 1 2	YES
? 0 1 2	YES
? 1 1 2	YES
+ 1 2 3	NO
? 4 3 1	
? 0 4 4	
? 4 1 4	

Problem F. K -я порядковая статистика на отрезке

Input file: kth.in
 Output file: kth.out
 Time limit: 6 секунд
 Memory limit: 512 мегабайт

Дан массив из N неотрицательных чисел, строго меньших 10^9 . Вам необходимо ответить на несколько запросов о величине k -й порядковой статистики на отрезке $[l, r]$.

Input

Первая строка содержит число N ($1 \leq N \leq 450\,000$) — размер массива.

Вторая строка может быть использована для генерации a_i — начальных значений элементов массива. Она содержит три числа a_1, l и m ($0 \leq a_1, l, m < 10^9$); для i от 2 до N

$$a_i = (a_{i-1} \cdot l + m) \bmod 10^9.$$

В частности, $0 \leq a_i < 10^9$.

Третья строка содержит одно целое число B ($1 \leq B \leq 1000$) — количество групп запросов.

Следующие B строк описывают одну группу запросов. Каждая группа запросов описывается 10 числами. Первое число G обозначает количество запросов в группе. Далее следуют числа x_1, l_x и m_x , затем y_1, l_y и m_y , затем, k_1, l_k и m_k ($1 \leq x_1 \leq y_1 \leq N, 1 \leq k_1 \leq y_1 - x_1 + 1, 0 \leq l_x, m_x, l_y, m_y, l_k, m_k < 10^9$). Эти числа используются для генерации вспомогательных последовательностей x_g и y_g , а также параметров запросов i_g, j_g и k_g ($1 \leq g \leq G$)

$$\begin{aligned} x_g &= ((i_{g-1} - 1) \cdot l_x + m_x) \bmod N + 1, & 2 \leq g \leq G \\ y_g &= ((j_{g-1} - 1) \cdot l_y + m_y) \bmod N + 1, & 2 \leq g \leq G \\ i_g &= \min(x_g, y_g), & 1 \leq g \leq G \\ j_g &= \max(x_g, y_g), & 1 \leq g \leq G \\ k_g &= (((k_{g-1} - 1) \cdot l_k + m_k) \bmod (j_g - i_g + 1)) + 1, & 2 \leq g \leq G \end{aligned}$$

Сгенерированные последовательности описывают запросы, g -й запрос состоит в поиске k_g -го по величине числа среди элементов отрезка $[i_g, j_g]$.

Суммарное количество запросов не превосходит 600 000.

Output

Выведите единственное число — сумму ответов на запросы.

Examples

kth.in	kth.out
5 1 1 1 5 1 1 0 0 3 0 0 2 0 0 1 2 0 0 5 0 0 3 0 0 1 1 0 0 5 0 0 5 0 0 1 3 0 0 3 0 0 1 0 0 1 1 0 0 4 0 0 1 0 0	15

Problem G. Менеджер памяти

Input file: `stdin`
Output file: `stdout`
Time limit: 8 секунды
Memory limit: 512 мегабайт

Одно из главных нововведений новейшей операционной системы Indows 7 — новый менеджер памяти. Он работает с массивом длины N и позволяет выполнять три самые современные операции:

- `copy(a, b, l)` — скопировать отрезок длины $[a, a + l - 1]$ в $[b, b + l - 1]$
- `sum(l, r)` — посчитать сумму элементов массива на отрезке $[l, r]$
- `print(l, r)` — напечатать элементы с l по r , включительно

Вы являетесь разработчиком своей операционной системы, и Вы, безусловно, не можете обойтись без инновационных технологий. Вам необходимо реализовать точно такой же менеджер памяти.

Input

Первая строка входного файла содержит целое число N ($1 \leq N \leq 1\,000\,000$) — размер массива, с которым будет работать Ваш менеджер памяти.

Во второй строке содержатся четыре числа $1 \leq X_1, A, B, M \leq 10^9 + 10$. С помощью них можно сгенерировать исходный массив чисел X_1, X_2, \dots, X_N . $X_{i+1} = (A * X_i + B) \bmod M$

Следующая строка входного файла содержит целое число K ($1 \leq K \leq 200\,000$) — количество запросов, которые необходимо выполнить Вашему менеджеру памяти.

Далее в K строках содержится описание запросов. Запросы заданы в формате:

- `cpy a b l` — для операции `copy`
- `sum l r` — для операции `sum` ($l \leq r$)
- `out l r` — для операции `print` ($l \leq r$)

Гарантируется, что суммарная длина запросов `print` не превышает 3 000. Также гарантируется, что все запросы корректны.

Output

Для каждого запроса `sum` или `print` выведите в выходной файл на отдельной строке результат запроса.

Examples

stdin	stdout
6	1 2 6 1 2 6
1 4 5 7	1 2 1 2 2 6
7	6
out 1 6	1 1 2 1 2 6
cpy 1 3 2	13
out 1 6	
sum 1 4	
cpy 1 2 4	
out 1 6	
sum 1 6	

Problem H. Persistent List

Input file: `stdin`
Output file: `stdout`
Time limit: 3 секунды
Memory limit: 512 мегабайт

Даны N списков. Каждый состоит из одного элемента.

Нужно научиться совершать следующие операции:

- **merge** — взять два каких-то уже существующих списка и породить новый, равный их конкатенации.
- **head** — взять какой-то уже существующий список L и породить два новых, в одном первый элемент L , во втором весь L кроме первого элемента.
- **tail** — взять какой-то уже существующий список L и породить два новых, в одном весь L кроме последнего элемента, во втором последний элемент L .

Для свежесозданных списков нужно говорить сумму элементов в них по модулю $10^9 + 7$.

Input

Число N ($1 \leq N \leq 10^5$). Далее N целых чисел от 1 до 10^9 — элементы списков. Исходные списки имеют номера $1, 2, \dots, N$.

Затем число M ($1 \leq M \leq 10^5$) — количество операций. Далее даны операции в следующем формате:

- `merge i j`
- `head i`
- `tail i`

Где i и j — номера уже существующих списков. Если в текущий момент имеется K списков, новый список получает номер $K + 1$.

Для операций **head** и **tail** считается, что сперва порождается левая часть, затем правая (см. пример). Также вам гарантируется, что никогда не будут рождаться пустые списки.

Output

Для каждого нового списка нужно вывести сумму элементов по модулю $10^9 + 7$.

Examples

stdin	stdout
4	3
1 2 3 4	7
7	10
merge 1 2	3
merge 3 4	7
merge 6 5	5
head 7	2
tail 9	5
merge 2 3	2
merge 1 1	

Problem I. Intercity Express

Input file:	стандартный ввод
Output file:	стандартный вывод
Time limit:	1.5 секунд
Memory limit:	256 мегабайт

Андрей разрабатывает систему для продажи железнодорожных билетов. Он собирается протестировать ее на Междугородней Экспресс линии, которая соединяет два больших города и имеет $n - 2$ промежуточных станций, то есть в итоге есть n станций, пронумерованных от 1 до n .

В Междугороднем Экспресс поезде есть s мест, пронумерованных с 1 до s . В тестирующем режиме система имеет доступ к базе данных, содержащей проданные билеты в направлении от станции 1 до станции n и должна отвечать на вопросы, можно ли продать билет от станции a до станции b , и если да, нужно найти минимальный номер места, которое свободно на протяжении всего пути между a и b .

Изначально система имеет только доступ на чтение, то есть даже если есть свободное место, она должна сообщить об этом, но не должна изменять данные.

Помогите Андрею протестировать его систему написанием программы, которые будет находить ответы на вопросы.

Input

Первая строка содержит число n — количество станций, s — количество мест и m — количество уже проданных билетов ($2 \leq n \leq 10^9$, $1 \leq s \leq 100\,000$, $0 \leq m \leq 100\,000$).

В следующих m строках описаны билеты, описание каждого билета состоит из трех чисел: c_i, a_i, b_i — номер места, которое занимает владелец билета, номер станции, с которой продан билет и номер станции, до которой продан билет ($1 \leq c_i \leq s$, $1 \leq a_i < b_i \leq n$).

Следующие строки содержат число q — количество запросов ($1 \leq q \leq 100\,000$). Специальное значение p должно поддерживаться в течение считывания запросов. Изначально $p = 0$.

Следующие q строк описывают запросы. Каждый запрос описывается двумя числами: x_i и y_i ($x_i < y_i$).

Чтобы получить города a и b , между которыми нужно проверить наличие места, используется следующая формула:

$a = x_i + p, b = y_i + p$. Ответ на запрос — число 0, если нет места на каждом отрезке между a и b , или минимальный номер свободного места. Гарантируется, что запрос корректен, то есть $1 \leq a < b \leq n$.

После ответа на запрос, надо приравнять число p полученному ответу на запрос.

Output

Для каждого запроса выведите ответ на него.

Examples

стандартный ввод	стандартный вывод
5 3 5	1
1 2 5	2
2 1 2	2
2 4 5	3
3 2 3	0
3 3 4	2
10	0
1 2	0
1 2	0
1 2	0
2 3	
-2 0	
2 4	
1 3	
1 4	
2 5	
1 5	

Note

Обратите внимание, что запросы выглядят так:

(1,2),(2,3),(3,4),(4,5),(1,3),(2,4),(3,5),(1,4),(2,5),(1,5).