Задача A. Куча

Имя входного файла:	`heap.in`
Имя выходного файла:	`heap.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайта

Сережа недавно прочитал реализацию структуры данных "куча" и написал ее на своем любимом языке программирования. Вот ее реализация на псевдокоде:

```
ExtractMin(H)
  result = H[1]
  H[1] = H[size[H]]
  size[H] = size[H] - 1
  PushDown(H, 1)
  return result

PushDown(H, i)
  child = index of minimum element among i, 2i, 2i+1
  if child != i
    swap(i, child)
    PushDown(child)

Add(H, value)
  size[H] = size[H] + 1
  H[size[H]] = value
  PushUp(H, size[H])

PushUp(H, i)
  if i > 1 and H[i/2] > H[i]
    swap(i, i/2)
    PushUp(i/2)
```

Теперь Сережа хочет проверить, справился ли он. Помогите ему.

Формат входных данных

Первая строка содержит число n — количество операций над кучей ($1 \leqslant n \leqslant 100\,000$). В следующих n строках заданы операции:

- `1 v` — применить операцию "ExtractMin" к версии номер v. Результатом этой операции является извлеченный элемент.
- `2 v t` — узнать, чему равно $H[t]$ в версии номер v. Результатом этой операции является значение $H[t]$.
- `3 v x` — добавить число x в версию номер v. Результатом этой операции является добавленный элемент. Гарантируется, что такого элемента нет в куче. $1 \leqslant x \leqslant 10^9$

В результате операции номер i ($1 \leqslant i \leqslant n$) новая версия кучи (возможно совпадающая с предыдущей) получает номер i.

Первоначально имеется пустая куча с номером ноль.

Формат выходных данных

Выведите результаты всех операций по одному в строке.

Примеры

heap.in	heap.out
18 3 0 4 3 1 2 3 2 1 3 3 3 2 4 1 2 4 2 2 4 3 2 4 4 1 4 2 9 1 2 9 2 2 9 3 1 9 2 13 1 2 13 2 1 13 2 16 1 1 16	4 2 1 3 1 3 2 4 1 2 3 4 2 3 4 3 4 4

Задача B. Менеджер памяти

Имя входного файла:	`memory.in`
Имя выходного файла:	`memory.out`
Ограничение по времени:	8 секунды
Ограничение по памяти:	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.

Формат входных данных

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

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

Следующая строка входного файла содержит целое число K ($1 \leqslant K \leqslant 200\,000$) — 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` ($l \leqslant r$)
- `out` l r — для операции `print` ($l \leqslant r$)

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

Формат выходных данных

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

Примеры

memory.in	memory.out
6 1 4 5 7 7 out 1 6 cpy 1 3 2 out 1 6 sum 1 4 cpy 1 2 4 out 1 6 sum 1 6	1 2 6 1 2 6 1 2 1 2 2 6 6 1 1 2 1 2 6 13

Задача C. Откат

Имя входного файла:	`rollback.in`
Имя выходного файла:	`rollback.out`
Ограничение по времени:	1.5 секунды
Ограничение по памяти:	256 мегабайта

Сергей работает системным администратором в очень крупной компании. Естественно, в круг его обязанностей входит резервное копирование информации, хранящейся на различных серверах и «откат» к предыдущей версии в случае возникновения проблем.

В данный момент Сергей борется с проблемой недостатка места для хранения информации для восстановления. Он решил перенести часть информации на новые сервера. К сожалению, если что-то случится во время переноса, он не сможет произвести откат, поэтому процедура переноса должна быть тщательно спланирована.

На данный момент у Сергея хранятся n точек восстановления различных серверов, пронумерованных от 1 до n. Точка восстановления с номером i позволяет произвести откат для сервера a_i. Сергей решил разбить перенос на этапы, при этом на каждом этапе в случае возникновения проблем будут доступны точки восстановления с номерами $l, l+1, \ldots, r$ для некоторых l и r.

Для того, чтобы спланировать перенос данных оптимальным образом, Сергею необходимо научиться отвечать на запросы: для заданного l, при каком минимальном r в процессе переноса будут доступны точки восстановления не менее чем k различных серверов.

Помогите Сергею.

Формат входных данных

Первая строка входного файла содержит два целых числа n и m, разделенные пробелами — количество точек восстановления и количество серверов ($1 \leqslant n, m \leqslant 100\,000$). Вторая строка содержит n целых чисел $a_1, a_2, \ldots, a_n$ — номера серверов, которым соответствуют точки восстановления ($1 \leqslant a_i \leqslant m$).

Третья строка входного файла содержит q — количество запросов, которые необходимо обработать ($1 \leqslant q \leqslant 100\,000$). В процессе обработки запросов необходимо поддерживать число p, исходно оно равно 0. Каждый запрос задается парой чисел x_i и y_i, используйте их для получения данных запроса следующим образом: $l_i = ((x_i + p) \bmod n) + 1$, $k_i = ((y_i + p) \bmod m) + 1$ ($1 \leqslant l_i, x_i \leqslant n$, $1 \leqslant k_i, y_i \leqslant m$). Пусть ответ на i-й запрос равен r. После выполнения этого запроса, следует присвоить p значение r.

Формат выходных данных

На каждый запрос выведите одно число — искомое минимальное r, либо 0, если такого r не существует.

Примеры

rollback.in	rollback.out
7 3 1 2 1 3 1 2 1 4 7 3 7 1 7 1 2 2	1 4 0 6

Задача D. Intercity Express

Имя входного файла:	`intercity.in`
Имя выходного файла:	`intercity.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	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.

Формат входных данных

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

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

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

Следующие $2q$ строк описывают запросы. Каждый запрос описывается двумя числами: x_i и y_i ($x_i \leqslant y_i$).

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.

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

Формат выходных данных

Для каждого запроса выведите ответ на него.

Примеры

intercity.in	intercity.out
5 3 5 1 2 5 2 1 2 2 4 5 3 2 3 3 3 4 10 1 2 1 2 1 2 2 3 -2 0 2 4 1 3 1 4 2 5 1 5	1 2 2 3 0 2 0 0 0 0

Замечание

Обратите внимание, что запросы выглядят так: $(1, 2)$, $(2, 3)$, $(3, 4)$, $(4, 5)$, $(1, 3)$, $(2, 4)$, $(3, 5)$, $(1, 4)$, $(2, 5)$, $(1, 5)$.

Задача E. Наборщик-рак

Имя входного файла:	`scrivener.in`
Имя выходного файла:	`scrivener.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Некоторые люди говорят, что Леонардо был большим почитателем Иоганна Гутенберга, немецкого кузнеца, который изобрел подвижную (наборную) печать, и что он воздал должное, сконструировав машину, названную им наборщик-рак — il gambero scrivano — очень простое наборное устройство. Оно чем-то похоже на современную простую пишущую машинку и выполняет всего 2 команды: одна, чтобы напечатать следующий символ, и вторая, чтобы отменить несколько последних команд. Замечательным свойством наборщика-рака является исключительная мощность команды отмены, которая рассматривается сама по себе как команда и тоже может быть отменена.

Вам необходимо реализовать программную модель наборщика-рака: она начинает работу с пустого текста, обрабатывает последовательность команд, передаваемых ей, и запросы относительно определенных позиций в текущем состоянии текста, как описано ниже.

- Напечатать букву L. Добавляет в конец текста один символ L — маленькую букву из диапазона `a` ... `z`.
- Отменить A команд. Отменяет последние A команд, где A — положительное целое число.
- Узнать символ на позиции P. Выводит символ — букву, находящуюся в позиции P текущего текста, где P — неотрицательное целое число. Первая буква текста имеет индекс 0. (Этот запрос не является командой и поэтому игнорируется командой отмены.)

Команда отмены отменяет предыдущие U команд в обратном порядке. Если отменяемая команда — это напечатать символ L, то из конца текста удаляется буква L. Если отменяемая команда — это отменить X команд, то для этого значения X она заново применяет предыдущие X команд в их оригинальном порядке.

Формат входных данных

В первой строке дано число n ($1 \leqslant n \leqslant 1\,000\,000$) — количество запросов.
В следующих n строках дано описание запросов. Запрос начинается с символа.

- Если этот символ равен `T`, то это запрос напечатать символ L, и далее в строке дан символ L.
- Если этот символ равен `U`, то это запрос отменить A команд, и далее в строке дано число A. Гарантируется, что A не будет превышать количество ранее полученных команд.
- Если этот символ равен `P`, то это запрос вывести символ на позиции P, и далее в строке дано число P. Символы в строке нумеруются с 0. Гарантируется, что P будет меньше чем текущая длинна текста (количество букв в текущем тексте).

Формат выходных данных

На одной строке выведите ответы на все запросы третьего типа.

Примеры

scrivener.in	scrivener.out
14 T a T b P 1 T d U 2 U 1 P 2 T e U 1 U 5 T c P 2 U 2 P 2	bdcd

Замечание

Пояснение к примеру:

Вызов	Результат	Текущий текст
T a		a
T b		ab
P 1	b	ab
T d		abd
U 2		a
U 1		abd
P 2	d	abd
T e		abde
U 1		abd
U 5		ab
T c		abc
P 2	c	abc
U 2		abd
P 2	d	abd