Задача A. Переворот

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

Дан массив. Надо научиться обрабатывать два типа запросов.

- `1 L R` - перевернуть отрезок [L, R]
- `2 L R` - найти минимум на отрезке [L, R]

Формат входных данных

Первая строка файла содержит два числа n, m. ($1 \leqslant n, m \leqslant 10^5$) Во второй строке находится n чисел a_i ($1 \leqslant a_i \leqslant 10^9$)- исходный массив. Остальные m строк содержат запросы, в формате описанном в условии. Для чисел L,R выполняется ограничение ($1 \leqslant L \leqslant R \leqslant n$).

Формат выходных данных

На каждый запрос типа 2, во входной файл выведите ответ на него, в отдельной строке.

Примеры

reverse.in	reverse.out
10 7 5 3 2 3 12 6 7 5 10 12 2 4 9 1 4 6 2 1 8 1 1 8 1 8 9 2 1 7 2 3 6	3 2 2 2

Задача B. Очередная

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Изначально вам дана перестановка чисел от 1 до N. Вам поступают запросы двух видов:

- 1 l_1 r_1 l_2 r_2 для выполнения требуется взять два подмассива нашей перестановки с границами $[l_1, r_1]$ и $[l_2, r_2]$ и поменять местами содержимое подмассивов друг с другом.
- 2 x найти место в перестановке, где находится число x и вывести 3 следующих за ним числа

Формат входных данных

В первой строке находится два числа N и Q — размер перестановки и общее количество запросов ($2 \leqslant N \leqslant 10000$, $1 \leqslant Q \leqslant 200000$). Во второй строке — перестановка чисел от одного до N. В следующих Q строках описаны запросы в виде либо 1 l_1 r_1 l_2 r_2($1 \leqslant l_1 \leqslant r_1 < l_2 \leqslant r_2 \leqslant N$, $r_1 - l_1 = r_2 - l_2$) либо 2 x ($1 \leqslant x \leqslant N$).

Формат выходных данных

Для каждого запроса второго типа выведите три числа — следующие числа за заданным, либо -1, если какого-то числа нет.

Примеры

стандартный ввод	стандартный вывод
6 6 1 2 3 4 5 6 2 4 1 1 2 4 5 2 4 2 1 1 1 3 4 6 2 1	5 6 -1 5 3 1 2 6 -1 2 6 4

Задача C. Своппер

Имя входного файла:	`swapper.in`
Имя выходного файла:	`swapper.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

Современные компьютеры зацикливаются
в десятки раз эффективнее человека

Рекламный проспект OS Vista-N

Перед возвращением в штаб-квартиру корпорации Аазу и Скиву пришлось заполнить на местной таможне декларацию о доходах за время визита. Получилась довольно внушительная последовательность чисел. Обработка этой последовательности заняла весьма долгое время.

— Своппер кривой, — со знанием дела сказал таможенник.

— А что такое своппер? — спросил любопытный Скив.

Ааз объяснил, что своппер — это структура данных, которая умеет делать следующее.

- Взять отрезок чётной длины от x до y и поменять местами число x с $x+1$, $x+2$ с $x+3$, и т.д.
- Посчитать сумму чисел на произвольном отрезке от a до b.

Учитывая, что обсчёт может затянуться надолго, корпорация «МИФ» попросила Вас решить проблему со своппером и промоделировать ЭТО эффективно.

Формат входных данных

Во входном файле заданы один или несколько тестов. В первой строке каждого теста записаны число N — длина последовательности и число M — число операций ($1 \leqslant N, M \leqslant 100\,000$). Во второй строке теста содержится N целых чисел, не превосходящих 10^6 по модулю — сама последовательность. Далее следуют M строк — запросы в формате `1` x_i y_i — запрос первого типа, и `2` a_i b_i — запрос второго типа. Сумма всех N и M по всему файлу не превосходит 200 000. Файл завершается строкой из двух нулей. Гарантируется, что $x_i < y_i$, а $a_i \leqslant b_i$.

Формат выходных данных

Для каждого теста выведите ответы на запросы второго типа, как показано в примере. Разделяйте ответы на тесты пустой строкой.

Примеры

swapper.in	swapper.out
5 5 1 2 3 4 5 1 2 5 2 2 4 1 1 4 2 1 3 2 4 4 0 0	Swapper 1: 10 9 2

Задача D. Река

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

Во Флатландии протекает богатая рыбой река Большой Флат. Много лет назад река была поделена между n рыболовными предприятиями, каждое из которых получило непрерывный отрезок реки. При этом i-е предприятие, если рассматривать их по порядку, начиная от истока, изначально получило отрезок реки длиной a_i.

С тех пор с рыболовными предприятиями во Флатландии k раз происходили различные события. Каждое из событий было одного из двух типов: банкротство некоторого предприятия или разделение некоторого предприятия на два. При некоторых событиях отрезок реки, принадлежащий предприятию, с которым это событие происходит, делится на две части. Каждый такой отрезок имеет длину большую или равную 2. Деление происходит по следующему правилу. Если отрезок имеет четную длину, то он делится на две равные части. Иначе он делится на две части, длины которых различаются ровно на единицу, при этом часть, которая ближе к истоку реки, имеет меньшую длину.

При банкротстве предприятия происходит следующее. Отрезок реки, принадлежавший обанкротившемуся предприятию, переходит к его соседям. Если у обанкротившегося предприятия один сосед, то этому соседу целиком передается отрезок реки обанкротившегося предприятия. Если же соседей двое, то отрезок реки делится на две части описанным выше способом, после чего каждый из соседей присоединяет к своему отрезку ближайшую к нему часть. При разделении предприятия отрезок реки, принадлежавший разделяемому предприятию, всегда делится на две части описанным выше способом. Разделившееся предприятие ликвидируется, и образуются два новых предприятия. Таким образом, после каждого события каждое предприятие владеет некоторым отрезком реки.

Министерство финансов Флатландии предлагает ввести налог на рыболовные предприятия, пропорциональный квадрату длины отрезка реки, принадлежащего соответствующему предприятию. Чтобы проанализировать, как будет работать этот налог, министр хочет по имеющимся данным узнать, как изменялась величина, равная сумме квадратов длин отрезков реки, принадлежащих предприятиям, после каждого произошедшего события.

Требуется написать программу, которая по заданному начальному разделению реки между предприятиями и списку событий, происходивших с предприятиями, определит, чему равна сумма квадратов длин отрезков реки, принадлежащих предприятиям, в начальный момент времени и после каждого события.

Формат входных данных

Первая строка входного файла содержит два целых числа: n и p — исходное количество предприятий ($2 \leqslant n \leqslant 100000$) и номер подзадачи ($0 \leqslant p \leqslant 4$) (считайте его просто так).

Вторая строка входного файла содержит n целых чисел $a_1, a_2, \ldots, a_n$ — длины исходных отрезков реки.

Третья строка входного файла содержит целое число k — количество событий, происходивших с предприятиями ($1 \leqslant k \leqslant 100000$).

Последующие k строк содержат описания событий, i-я строка содержит два целых числа: e_i и v_i — тип события и номер предприятия, с которым оно произошло. Значение $e_i = 1$ означает, что предприятие, которое после всех предыдущих событий является v_i-м по порядку, если считать с единицы от истока реки, обанкротилось, а значение $e_i = 2$ означает, что это предприятие разделилось на два.

Гарантируется, что значение v_i не превышает текущее количество предприятий. Гарантируется, что если отрезок предприятия при банкротстве или разделении требуется поделить на две части, то он имеет длину большую или равную 2. Гарантируется, что если на реке осталось единственное предприятие, оно не банкротится.

Формат выходных данных

Выходной файл должен содержать $(k + 1)$ целых чисел, по одному в строке. Первая строка должна содержать исходную сумму квадратов длин отрезков реки, а каждая из последующих k строк — сумму квадратов длин отрезков реки после очередного события.

Примеры

river.in	river.out
4 0 3 5 5 4 5 1 1 2 1 1 3 2 2 1 3	75 105 73 101 83 113

Задача E. Вперёд!

Имя входного файла:	`movetofront.in`
Имя выходного файла:	`movetofront.out`
Ограничение по времени:	3 секунды
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 которых 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 $2, 3, 6, 1, 5, 4$, то после приказа «Рядовые с 2 по 4 — 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 приказов Капрала. Найдите порядок, в котором будут стоять солдаты после исполнения всех приказов.

Формат входных данных

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

Формат выходных данных

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

Примеры

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

Задача F. Неявный Ключ

Имя входного файла:	`implicitkey.in`
Имя выходного файла:	`implicitkey.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Научитесь быстро делать две операции с массивом:
- `add i x` — добавить после i-го элемента x ($0 \leqslant i \leqslant n$)
- `del i` — удалить i-й элемент ($1 \leqslant i \leqslant n$)

Формат входных данных

На первой строке n_0 и m ($1 \leqslant n_0, m \leqslant 10^5$) — длина исходного массива и количество запросов. На второй строке n_0 целых чисел от 0 до $10^9 - 1$ — исходный массив. Далее m строк, содержащие запросы. Гарантируется, что запросы корректны: например, если просят удалить i-й элемент, он точно есть.

Формат выходных данных

Выведите конечное состояние массива. На первой строке количество элементов, на второй строке сам массив.

Примеры

implicitkey.in	implicitkey.out
3 4 1 2 3 del 3 add 0 9 add 3 8 del 2	3 9 2 8