

## Задача А. Вперёд!

Имя входного файла: `movetofront.in`  
Имя выходного файла: `movetofront.out`  
Ограничение по времени: 1 секунда  
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  $m$  ( $2 \leq n \leq 100\,000$ ,  $1 \leq m \leq 100\,000$ ) — число солдат и число приказов. Следующие  $m$  строк содержат приказы в виде двух целых чисел:  $l_i$  и  $r_i$  ( $1 \leq l_i \leq r_i \leq n$ ).

### Формат выходных данных

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

### Примеры

<code>movetofront.in</code>	<code>movetofront.out</code>
6 3	1 4 5 2 3 6
2 4	
3 5	
2 2	

## Задача В. 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 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  $l \leq x \leq r$ .

### 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 «?  $l$   $r$ » задаёт запрос  $sum(l, r)$ .

Если операция «+  $i$ » идёт во входном файле в начале или после другой операции «+», то она задаёт операцию  $add(i)$ . Если же она идёт 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.

### Примеры

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

## Задача С. 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  $n, m$ . ( $1 \leq n, m \leq 10^5$ ) Во второй строке находится  $n$  чисел  $a_i$  ( $1 \leq a_i \leq 10^9$ )- исходный массив. Остальные  $m$  строк содержат запросы, в формате описанном в условии. Для чисел L,R выполняется ограничение ( $1 \leq L \leq R \leq n$ ).

### Формат выходных данных

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

### Примеры

<code>reverse.in</code>	<code>reverse.out</code>
10 7	3
5 3 2 3 12 6 7 5 10 12	2
2 4 9	2
1 4 6	2
2 1 8	
1 1 8	
1 8 9	
2 1 7	
2 3 6	

## Задача D. Мобильные телефоны

Имя входного файла: `mobile-phone.in`  
Имя выходного файла: `mobile-phone.out`  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 64 мегабайта

В области Тампере существуют станции обслуживания мобильных телефонов четвертого поколения, которые работают следующим образом. Вся область разделена на квадраты. Квадраты формируют матрицу  $S \times S$ , строки и столбы которой нумеруются с 0 до  $S - 1$ . Каждый квадрат содержит одну станцию обслуживания. Количество активных мобильных телефонов внутри квадрата может меняться, потому что телефон может перемещаться из одного квадрата в другой или выключаться. Временами каждая станция обслуживания передает сведения об изменении в количестве активных телефонов на главную станцию обслуживания (вместе с номером строки и столбца матрицы).

Напишите программу, которая принимает эти сведения и отвечает на запросы о текущем количестве активных мобильных телефонов в любой прямоугольной области.

### Формат входных данных

Каждый запрос к программе задан на отдельной строке и состоит из одного целого числа, описывающего команду, и последовательности чисел, являющиеся параметрами команды.

Команда	Параметры	Значение
0	$S$	Инициализирует матрицу размером $S \times S$ , состоящую только из нулей. Эта команда поступает один раз и является самой первой командой.
1	$X Y A$	Добавить число $A$ к количеству активных мобильных телефонов в квадрате $(X; Y)$ . Число $A$ может быть как положительным, так и отрицательным.
2	$L B R T$	Выдать текущую сумму количеств активных мобильных телефонов во всех квадратах $(X; Y)$ , таких что $L \leq X \leq R, B \leq Y \leq T$ .
3	$\emptyset$	Означает конец входных данных. Эта команда поступает ровно один раз и является самой последней командой.

$S$  не превосходит 1024, количество команд — 60 002, значение ячейки в любой момент времени —  $2^{15}$ , количество активных телефонов во всей таблице —  $2^{30}$ . Величина обновления не превосходит по модулю  $2^{15}$ , также ни в один момент времени число активных телефонов в квадрате не становится меньше нуля.

Все индексы нумеруются начиная с нуля.

### Формат выходных данных

На запросы типов 0, 1, 3 отвечать не надо. Для каждого запроса второго типа выведите строку, содержащую одно целое число — ответ на него.

### Примеры

<code>mobile-phone.in</code>	<code>mobile-phone.out</code>
0 4	3
1 1 2 3	4
2 0 0 2 2	
1 1 1 2	
1 1 2 -1	
2 1 1 2 3	
3	

## Задача Е. Сwoппер

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.

— Сwoппер кривой, — со знанием дела сказал таможенник.

— А что такое сwoппер? — спросил любопытный Скив.

Ааз объяснил, что сwoппер — 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 сwoппером 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 ( $1 \leq N, M \leq 100\,000$ ). 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  $a_i \leq b_i$ .

### Формат выходных данных

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

### Примеры

<code>swapper.in</code>	<code>swapper.out</code>
5 5	Swapper 1:
1 2 3 4 5	10
1 2 5	9
2 2 4	2
1 1 4	
2 1 3	
2 4 4	
0 0	