

Задача А. Очередь

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

Реализуйте работу очереди. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо “+ N”, либо “-”. Команда “+ N” означает добавление в очередь числа N , по модулю не превышающего 10^9 . Команда “-” означает изъятие элемента из очереди.

Формат входных данных

В первой строке содержится количество команд — M ($1 \leq M \leq 10^6$). В последующих строках содержатся команды, по одной в каждой строке.

Формат выходных данных

Выведите числа, которые удаляются из очереди, по одному в каждой строке. Гарантируется, что изъятий из пустой очереди не производится.

Примеры

<code>queue.in</code>	<code>queue.out</code>
4	1
+ 1	10
+ 10	
-	
-	

Задача В. Игра в пьяницу

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

В игре в пьяницу карточная колода раздаётся поровну двум игрокам. Далее они вскрывают по одной верхней карте, и тот, чья карта старше, забирает себе обе вскрытые карты, которые кладутся под низ его колоды. Тот, кто остаётся без карт — проигрывает.

Для простоты будем считать, что все карты различны по номиналу, а также, что самая младшая карта побеждает самую старшую карту («шестерка берет туза»).

Игрок, который забирает себе карты, сначала кладёт под низ своей колоды карту первого игрока, затем карту второго игрока (то есть карта второго игрока оказывается внизу колоды).

Напишите программу, которая моделирует игру в пьяницу и определяет, кто выигрывает. В игре участвует n карт, имеющих значения от 0 до $n - 1$, большая карта побеждает меньшую, карта со значением 0 побеждает карту $n - 1$.

Формат входных данных

Программа получает на вход три строки. В первой строке содержится целое чётное число n ($2 \leq n \leq 100\,000$). Вторая строка содержит $\frac{n}{2}$ чисел — карты первого игрока, а третья — $\frac{n}{2}$ карт второго игрока. Карты перечислены сверху вниз, то есть каждая строка начинается с той карты, которая будет открыта первой. Гарантируется, что каждая из карт встречается в колодах игроков ровно один раз.

Формат выходных данных

Программа должна определить, кто выигрывает при данной раздаче, и вывести слово «first» или «second», после чего вывести количество ходов, сделанных до выигрыша. Если на протяжении $2 \cdot 10^5$ ходов игра не заканчивается, программа должна вывести слово «draw».

Примеры

<code>card-game.in</code>	<code>card-game.out</code>
10	second 5
1 3 5 7 9	
2 4 6 8 0	

Задача С. Постфиксная запись

Имя входного файла: `postfix.in`
Имя выходного файла: `postfix.out`
Ограничение по времени: 1 second
Ограничение по памяти: 64 megabytes

В постфиксной записи (или обратной польской записи) операция записывается после двух операндов. Например, сумма двух чисел A и B записывается как $A B +$. Запись $B C + D *$ обозначает привычное нам $(B + C) * D$, а запись $A B C + D * +$ означает $A + (B + C) * D$. Достоинство постфиксной записи в том, что она не требует скобок и дополнительных соглашений о приоритете операторов для своего чтения.

Дано выражение в обратной польской записи. Определите его значение.

Формат входных данных

В единственной строке записано выражение в постфиксной записи, содержащее однозначные числа и операции $+$, $-$, $*$. Строка содержит не более 100 чисел и операций.

Формат выходных данных

Необходимо вывести значение записанного выражения. Гарантируется, что результат выражения, а также результаты всех промежуточных вычислений по модулю меньше 2^{31} .

Примеры

<code>postfix.in</code>	<code>postfix.out</code>
8 9 + 1 7 - *	-102

Задача D. Минимум на стеке

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

Вам требуется реализовать структуру данных, выполняющую следующие операции:

1. Добавить элемент x в конец структуры.
2. Удалить последний элемент из структуры.
3. Выдать минимальный элемент в структуре.

Формат входных данных

В первой строке входного файла задано одно целое число n — количество операций ($1 \leq n \leq 10^6$). В следующих n строках заданы сами операции. В i -ой строке число t_i — тип операции (1, если операция добавления, 2, если операция удаления, 3, если операция минимума). Если задана операция добавления, то через пробел записано целое число x — элемент, который следует добавить в структуру ($-10^9 \leq x \leq 10^9$). Гарантируется, что перед каждой операцией удаления или нахождения минимума структура не пуста.

Формат выходных данных

Для каждой операции нахождения минимума выведите одно число — минимальный элемент в структуре. Ответы разделяйте переводом строки.

Примеры

<code>stack.in</code>	<code>stack.out</code>
8	-3
1 2	2
1 3	2
1 -3	
3	
2	
3	
2	
3	

Задача E. Очередь

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

Гоблины Мглистых гор очень любят ходить к своим шаманам. Так как гоблинов много, к шаманам часто образуются очень длинные очереди. А поскольку много гоблинов в одном месте быстро образуют шумную толпу, которая мешает шаманам производить сложные

медицинские манипуляции, последние решили установить некоторые правила касательно порядка в очереди.

Обычные гоблины при посещении шаманов должны вставать в конец очереди. Привилегированные же гоблины, знающие особый пароль, встают ровно в ее середину, причем при нечетной длине очереди они встают сразу за центром.

Так как гоблины также широко известны своим непочтительным отношением ко всяческим правилам и законам, шаманы попросили вас написать программу, которая бы отслеживала порядок гоблинов в очереди.

Формат входных данных

В первой строке записано одно целое число N ($1 \leq N \leq 10^5$) — число запросов к вашей программе. В следующих N строках заданы описания запросов в следующем формате:

- «+ i » — к очереди присоединяется гoblin i ($1 \leq i \leq N$) и встает в ее конец;
- «* i » — привилегированный гoblin i встает в середину очереди ($1 \leq i \leq N$);
- «-» — гoblin выходит из очереди и заходит к шаманам. Гарантируется, что на момент каждого такого запроса очередь будет не пуста.

Формат выходных данных

Для каждого запроса третьего типа в отдельной строке выведите номер гоблина, который должен зайти к шаманам.

Примеры

<code>queue.in</code>	<code>queue.out</code>
7	1
+ 1	2
+ 2	3
-	
+ 3	
+ 4	
-	
-	
10	1
+ 1	3
+ 2	2
* 3	5
-	4
+ 4	
* 5	
-	
-	
-	
-	