Задача A. Реализуйте стек

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Требуется реализовать стек с двумя операциями.

«Первая» операция добавляет элемент в стек, а «вторая» — удаляет. На каждую «вторую» операцию необходимо вывести удаленное число. Гарантируется, что всегда есть, что удалять.

Добавление элемента, называемое также проталкиванием (push), возможно только в вершину стека (добавленный элемент становится первым сверху). Удаление элемента, называемое также выталкивание (pop), возможно также только из вершины стека, при этом, второй сверху элемент становится верхним.

Формат входных данных

В первой строчке находится число операций N ($1 \leqslant N \leqslant 100\,000$). В следующих N строчках первое число — номер операции, второе (только для «первой» операции) — добавляемое число (это число натуральное и не превосходит 100 000).

Формат выходных данных

Выведите все удаленные числа по одному в строке.

Пример

стандартный ввод	стандартный вывод
6 1 3 1 4 1 6 2 1 8 2	6 8

Задача B. Очередь

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

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 ($1 \leqslant M \leqslant 10^6$). 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.

Пример

стандартный ввод	стандартный вывод
4 + 1 + 10 - -	1 10

Задача C. Хипуй!

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В этой задаче вам необходимо организовать структуру данных Heap для хранения целых чисел, над которой определены следующие операции:

- `Insert(X)` — добавить в Heap число X;
- `Extract` — достать из Heap наибольшее число (удалив его при этом).

Эту задачу нужно решить без использования встроенных структур данных для поиска максимального числа.

Формат входных данных

Во входном файле записано количество команд N ($1 \leqslant N \leqslant 100\,000$), потом последовательность из N команд, каждая в своей строке.

Каждая команда имеет такой формат: „`0 <число>`“ или „`1`“, что означает соответственно операции `Insert(<число>)` и `Extract`. Добавляемые числа находятся в интервале от 1 до 10^7 включительно.

Гарантируется, что при выполнении команды `Extract` в структуре находится по крайней мере один элемент.

Формат выходных данных

В выходной файл для каждой команды извлечения необходимо вывести число, полученное при выполнении команды `Extract`.

Пример

стандартный ввод	стандартный вывод
7 0 100 0 10 1 0 5 0 30 0 50 1	100 50

Задача D. Игра в пьяницу

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

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 ($2 \leqslant n \leqslant 100\,000$). 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`».

Пример

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

Задача E. Скобки

Имя входного файла:	`brackets.in`
Имя выходного файла:	`brackets.out`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	256 мегабайт

Требуется определить, является ли правильной данная последовательность круглых, квадратных и фигурных скобок.

Формат входных данных

В единственной строке входного файла записано подряд N скобок ($1 \leqslant N \leqslant 10^5$).

Формат выходных данных

В выходной файл вывести «`YES`», если данная последовательность является правильной, и «`NO`» в противном случае.

Примеры

brackets.in	brackets.out
`()`	`YES`
`([])`	`YES`

Замечание

Скобочная последовательность называется правильной, если ее можно получить из какого-либо математического выражения вычеркиванием всех символов, кроме скобок.

Формальное определение правильной скобочной последовательности таково: 1. Пустая последовательность является правильной. 2. Если A – правильная скобочная последовательность, то (A), [A] и A – правильные скобочные последовательности. 3. Если A и B – правильные скобочные последовательности, то AB – правильная скобочная последовательность.

Задача F. Парикмахерская

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

В парикмахерской работает один мастер. Он тратит на одного клиента ровно 20 минут, а затем сразу переходит к следующему, если в очереди кто-то есть, либо ожидает, когда придет следующий клиент.

Даны времена прихода клиентов в парикмахерскую (в том порядке, в котором они приходили).

Также у каждого клиента есть характеристика, называемая *степенью нетерпения*. Она показывает, сколько человек может максимально находиться в очереди перед клиентом, чтобы он дождался своей очереди и не ушел раньше. Если в момент прихода клиента в очереди находится больше людей, чем степень его нетерпения, то он решает не ждать своей очереди и уходит. Клиент, который обслуживается в данный момент, также считается находящимся в очереди.

Требуется для каждого клиента указать время его выхода из парикмахерской.

Формат входных данных

В первой строке вводится натуральное число N, не превышающее 100 — количество клиентов.

В следующих N строках вводятся времена прихода клиентов — по два числа, обозначающие часы и минуты (часы — от 0 до 23, минуты — от 0 до 59) и степень его нетерпения (неотрицательное целое число не большее 100) — максимальное количество человек, которое он готов ждать впереди себя в очереди. Времена указаны в порядке возрастания (все времена различны).

Гарантируется, что всех клиентов успеют обслужить до полуночи.

Если для каких-то клиентов время окончания обслуживания одного клиента и время прихода другого совпадают, то можно считать, что в начале заканчивается обслуживание первого клиента, а потом приходит второй клиент.

Формат выходных данных

В выходной файл выведите N пар чисел: времена выхода из парикмахерской 1-го, 2-го, ..., N-го клиента (часы и минуты). Если на момент прихода клиента человек в очереди больше, чем степень его нетерпения, то можно считать, что время его ухода равно времени прихода.

Пример

saloon.in	saloon.out
3 10 0 0 10 1 1 10 2 1	10 20 10 40 10 2

Задача G. Минимум и максимум

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

Пусть есть множество целых чисел. Необходимо реализовать структуру данных для их хранения, поддерживающую следующие операции: `GetMin` — извлечение минимума, `GetMax` — извлечение максимума, `Insert(N)` — добавление числа в множество.

Формат входных данных

В первой строке входного файла записано одно целое число N ($1 \leqslant N \leqslant 100\,000$) — число запросов к структуре. Затем в N строках следуют запросы по одному в строке: `GetMin`, `GetMax`, `Insert(A)` — извлечение минимума, максимума и добавление числа A ($1 \leqslant A \leqslant 2^{31} - 1$). Запросы корректны, то есть нет операций извлечения для пустого множества.

Формат выходных данных

Для каждого запроса `GetMin` или `GetMax` выведите то число, которое было извлечено.

Пример

minmax.in	minmax.out
10 Insert(100) Insert(99) Insert(1) Insert(2) GetMin GetMax Insert(1) GetMin GetMin GetMax	1 100 1 2 99

Задача N. Бэнг

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Роботы, как и люди, любят играть в настольные игры. Но в отличие от нас, им не обязательно собираться вместе за одним столом. Ведь роботы могут поключиться к одному серверу, и синхронизируясь, визуализировать игру так, будто они сидят все вместе.

Так вот, n роботов решили сыграть в игру «Бэнг». Суть игры в том, что игроки сидят в кругу, у i-го робота есть h_i жизней и некоторое количество патронов. Во время хода, игрок может "стрелять"в другого игрока, тем самым отнимать жизни, теряя соответствующее отнятым жизням количество патронов. Но "стрелять"можно в любого из k по правую или по левую сторону от вас игроков. Если у кого-либо число жизней становится равным нулю, данный игрок выходит из-за стола.

Роботу Джеки стало интересно, какое минимальное кол-во патронов ему необходимо, чтобы за свой ход выкинуть из-за стола робота Чарли, если он может за ход "стрелять" по очереди в нескольких роботов.

Каждый робот пронумерован против часовой стрелки, начиная от Джеки. Получается, Джеки имеет номер 1. Справа от него сидит робот с номером 2, слева — робот с номером n. Чарли сидит под номером m.

Формат входных данных

В первой строке входных файлов содержится три разделенных пробелом целых числа n, k, m ($2 \leqslant n \leqslant 100\,000$, $1 \leqslant k < n$, $2 \leqslant m \leqslant n$).

Во второй строке содержится n разделенных пробелом чисел, где на i-й позиции стоит h_i ($1 \leqslant h_i \leqslant 10000$) - количество жизней i-го игрока.

Формат выходных данных

Выведите одно число — минимальное количество патронов, необходимое для Джеки, чтобы за свой ход выкинуть из-за стола робота Чарли.

Примеры

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

Замечание

В первом примере Джеки не может "стрелять"сразу в Чарли, поэтому для начала ему нужно потратить 2 патрона, чтобы выкинуть из-за стола робота под номером 3. После этого Джеки уже может выкинуть Чарли, потратив 5 патронов.

Во втором примере Джеки сразу может поратить 6 патронов и выкинуть Чарли из-за стола.

Задача I. Постфиксная запись

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

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}.

Пример

postfix.in	postfix.out
8 9 + 1 7 - *	-102

Задача J. Астроград

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

В Астрополисе прошел концерт популярной группы Астроград. За пару дней до концерта перед кассой выстроилась огромная очередь из людей, желающих туда попасть. Изначально очередь была пуста. В каждый из n моментов времени происходило следующее:

1. В очередь пришел новый человек с уникальным номером id, он встает в очередь последним.
2. Человеку, стоящему спереди очереди, удалось купить билет. Он уходит.
3. Человеку, стоящему последнему в очереди, надоело ждать. Он уходит.
4. Человек с уникальным номером q хочет знать, сколько людей стоит в очереди спереди него.
5. Очередь хочет знать, человек с каким уникальным номером стоит сейчас первым и задерживает всех.

Вам необходимо написать программу, которая умеет обрабатывать описанные события.

Формат входных данных

В первой строке дано целое число n ($1 \leqslant n \leqslant 10^5$) — количество событий. В каждой из следующих n строк дано описание событий: номер события, а также число id ($1 \leqslant id \leqslant 10^5$) для событий типа 1 и число q для событий типа 4. События происходили в том порядке, в каком они описаны во входном файле. Гарантируется корректность всех событий.

Формат выходных данных

Выведите ответы для событий типа 4 и 5 в том порядке, в каком они описаны во входном файле.

Пример

astrograd.in	astrograd.out
7 1 1 5 1 3 3 2 1 2 4 2	1 0

Замечание

В примере из условия происходили следующие события:

1. В очередь пришел человек с $id = 1$. Очередь: `[ 1 ]`
2. Первым в очереди стоит человек с $id = 1$. Очередь: `[ 1 ]`
3. В очередь пришел человек с $id = 3$. Очередь: `[ 1, 3 ]`
4. Последнему в очереди надоело стоять и он уходит. Очередь: `[ 1 ]`
5. Первому в очереди удалось купить билет и он уходит. Очередь: `[ ]`
6. В очередь пришел человек с $id = 2$. Очередь: `[ 2 ]`
7. $q = 2$ хочет знать, сколько человек стоит перед ним. Очередь: `[ 2 ]`