Задача A. Хип ли?

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

Структуру данных Heap можно реализовать на основе массива.

Для этого должно выполнятся *основное свойство Heap'a*, которое заключается в следующем. Для каждого $1 \leqslant i \leqslant n$ выполняются следующие условия:

- Если $2i \leqslant n$, то $a[i] \leqslant a[2i]$
- Если $2i + 1 \leqslant n$, то $a[i] \leqslant a[2i + 1]$

Дан массив целых чисел. Определите является ли он Heap'ом.

Формат входных данных

Первая строка входного файла содержит целое число n ($1 \leqslant n \leqslant 10^5$). Вторая строка содержит n целых чисел по модулю не превосходящих $2 \cdot 10^9$.

Формат выходных данных

Выведите «`YES`», если массив является Heap'ом и «`NO`» в противном случае.

Примеры

isheap.in	isheap.out
5 1 0 1 2 0	NO
5 1 3 2 5 4	YES

Задача B. Скобки

Имя входного файла:	`brackets.in`
Имя выходного файла:	`brackets.out`
Ограничение по времени:	0.5 second
Ограничение по памяти:	64 megabytes

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 C. Парикмахерская

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

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.

Примеры

saloon.in	saloon.out
3 10 0 0 10 1 1 10 2 1	10 20 10 40 10 2
5 1 0 100 2 0 0 2 1 0 2 2 3 2 3 0	1 20 2 20 2 1 2 40 2 3

Задача D. Великое Лайнландское переселение

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Лайнландия представляет из себя одномерный мир, являющийся прямой, на котором располагаются N городов, последовательно пронумерованных от 0 до N - 1. Направление в сторону от первого города к нулевому названо западным, а в обратную — восточным.

Когда в Лайнландии неожиданно начался кризис, все были жители мира стали испытывать глубокое смятение. По всей Лайнландии стали ходить слухи, что на востоке живётся лучше, чем на западе.

Так и началось Великое Лайнландское переселение. Обитатели мира целыми городами отправились на восток, покинув родные улицы, и двигались до тех пор, пока не приходили в город, в котором средняя цена проживания была меньше, чем в родном.

Формат входных данных

В первой строке дано число N ($2 \leqslant N \leqslant 10^5$) - количество городов в Лайнландии. Во второй строке дано N чисел a_i ($0 \leqslant a_i \leqslant 10^9$) - средняя цена проживания в городах с нулевого по (N - 1)-ый соответственно.

Формат выходных данных

Для каждого города в порядке с нулевого по (N - 1)-ый выведите номер города, в который переселятся его изначальные жители. Если жители города не остановятся в каком-либо другом городе, отправившись в Восточное Бесконечное Ничто, выведите -1.

Примеры

стандартный ввод	стандартный вывод
10 1 2 3 2 1 4 2 5 3 1	-1 4 3 4 -1 6 9 8 9 -1

Задача E. Гемоглобин

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	64 мегабайта

Каждый день к Грегори Хаусу приходит много больных, и у каждого измеряется уровень гемоглобина в крови. Данные по всем пациентам заносятся в базу данных.

Но волчанка попадается один раз на миллион, а работать с остальными неинтересно. Чтобы Хаус не выгонял больных, Кадди иногда запрашивает статистику по k последним больным: ей хочется знать сумму их уровня гемоглобина.

Также Хаус — мизантроп: он смотрит уровень гемоглобина больного, который поступил к нему позже всех, и, видя, что это точно не волчанка, выписывает его из больницы и удаляет информацию о нем из базы.

Автоматизацию процесса Хаус поручил Чейзу. Но Чейз почему-то не справился с этой задачей и попросил вас ему помочь.

Формат входных данных

Первой строкой входного файла задано число n ($1 \leqslant n \leqslant 100000$) - число обращений к базе данных. Запросы к базе выглядят следующим образом: „+x” ($1 \leqslant x \leqslant 10^9$) - добавить пациента с уровнем гемоглобина x в базу, „-” - удалить последнего пациента из базы, „?k” ($1 \leqslant k \leqslant 100000$) - вывести суммарный гемоглобин последних k пациентов. Гарантируется, что k не превосходит число элементов в базе. Также гарантируется, что запросов на удаление к пустой базе не поступает. Перед началом работы база данных пуста.

Формат выходных данных

Для каждого запроса „-” вывести уровень гемоглобина в крови пациента, а для каждого запроса „?k” — суммарный гемоглобин у последних k поступивших пациентов. Ответы выводите в порядке поступления запросов.

Примеры

стандартный ввод	стандартный вывод
7 +1 +2 +3 ?2 - - ?1	5 3 2 1

Задача F. Гоблины и шаманы

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

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 толку, которая мешает шаманам проводить 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 входных данный записано число N ($1 \leqslant N \leqslant 10^5$) - количество запросов к программе. Следующие N строк содержат описание запросов в формате:

- „+ i” - гоблин с номером i ($1 \leqslant i \leqslant N$) встает в конец очереди.
- „* i” - привилегированный гоблин с номером i встает в середину очереди.
- „-” - первый гоблин из очереди уходит к шаманам. Гарантируется, что на момент такого запроса очередь не пуста.

Формат выходных данных

Для каждого запроса типа „-” программа должна вывести номер гоблина, который должен зайти к шаманам.

Примеры

стандартный ввод	стандартный вывод
7 + 1 + 2 - + 3 + 4 - -	1 2 3

Задача G. Электрички

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

На вокзале есть K тупиков, куда прибывают электрички. Этот вокзал является их конечной станцией, поэтому электрички, прибыв, некоторое время стоят на вокзале, а потом отправляются в новый рейс (в ту сторону, откуда прибыли).

Дано расписание движения, в котором указаны события прибытия и отбытия для каждой из электричек в хронологическом порядке. Поскольку вокзал — конечная станция, то электричка может стоять на нем довольно долго, в частности, электричка, которая прибывает раньше другой, отправляться обратно может значительно позднее.

Тупики пронумерованы числами от 1 до K. Когда электричка прибывает, ее ставят в свободный тупик с минимальным номером.

Напишите программу, которая по данному расписанию для каждой электрички определит номер тупика, куда прибудет эта электричка.

Формат входных данных

В первой строке вводится число K — количество тупиков ($1 \leqslant K \leqslant 20000$). Далее следуют строки, описывающие события прибытия/отбытия электричек. Каждая электричка задаётся номером своей противоположной конечной станции — числом $a_i (0 \leqslant a_i \leqslant 100000)$. Событие $+ a_i$ означает, что прибывает электричка из города номер a_i, событие $- a_i$ — что эта электричка отправляется обратно. Общее количество электричек, фигурирующих в условии — не более 100000, для каждой фигурирующей электрички присутствуют оба события.

Считается, что в нулевой момент времени все тупики на вокзале свободны.

Формат выходных данных

Выведите по одному числу на каждую электричку — номер тупика, куда её поставят по прибытии. Если тупиков не достаточно для того, чтобы организовать движение электричек согласно расписанию, выведите два числа: первое должно равняться 0 (нулю), а второе содержать номер города первой из электричек, которая не сможет прибыть на вокзал.

Примеры

trains.in	trains.out
3 + 0 + 1 - 0 + 2 - 2 + 3 + 4 - 1 - 3 - 4	0 1 1 2 2 1 3 1 4 3
2 + 0 + 1 + 2 - 1 - 0 - 2	0 2

Задача H. Дело в шляпах

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

Как-то раз в город Шляп заехал известный парикмахер. До его приезда парикмахеры были явно не очень, так как все жители города предпочитали ходить в шляпах. Но наконец-то настало время снять шляпы!

Парикмахер открыл свою временную парикмахерскую и работает без остановок, пока есть посетители. Жители приходят к нему в тот момент времени, когда им это удобно, и становятся в очередь. Каждому из них требуется своё время на создание индивидуальной стрижки. Парикмахер зовёт первого человека в порядке очереди, стрижёт его, и после ухода посетителя сразу зовёт следующего.

Стоять в очереди скучно, поэтому если подряд приходят двое или более людей в шляпах одинакового фасона — они начинают между собой активно общаться и необычайно гордиться своими шляпами (но всё равно заходят на стрижку, если уж их очередь подошла). Однако, если следом за ними в очередь встаёт человек в шляпе другого фасона, то вся группа подряд стоящих людей в одинаковых шляпах подозрительно смотрит на только что пришедшего «чужого» и совсем уходит из очереди. При этом очередь сдвигается и может появиться новая группа общающихся людей.

Так как обсуждение одинаковых шляп — это очень интересная тема, появление «чужого» человека в очереди привлекает внимание группы сильнее, чем парикмахер. Поэтому если одновременно пришёл человек в другой шляпе и парикмахер зовёт следующего — вся группа уходит, даже если один из них должен был сейчас зайти на стрижку. К парикмахеру при этом зайдёт следующий из оставшейся очереди, возможно даже только что пришедший «чужой».

Местного шляпника теперь интересует, каким жителям ему больше не нужно будет делать шляпы, так как они будут ходить с новыми стильными стрижками?

Формат входных данных

В первой строке содержится число N ($1 \leqslant N \leqslant 10^5$) — количество людей, которые придут к парикмахеру.

Каждая из следующих N строк обозначает пришедшего к парикмахеру жителя и содержит по три числа: фасон шляпы (все фасоны местного шляпника пронумерованы от 1 до 10^9), момент времени прихода s ($1 \leqslant s \leqslant 10^9$), и время на стрижку t ($1 \leqslant t \leqslant 10^9$). Строки упорядочены по времени прихода жителей. Гарантируется, что все приходят в разное время.

Так как парикмахер очень крут, гарантируется, что он успеет постричь всех жителей до момента времени $2 \cdot 10^9$, даже если бы из очереди никто не уходил.

Формат выходных данных

В единственной строке выведите через пробел номера людей в очереди в порядке возрастания, которых парикмахер всё-таки пострижёт. Люди нумеруются в порядке прихода в очередь, начиная с 1.

Примеры

hats.in	hats.out
5 1 2 7 2 4 3 2 6 2 1 7 3 3 8 2	1 4 5