Задача A. Шифровка

Имя входного файла: `decode.in`
Имя выходного файла: `decode.out`

Мюллер много раз пытался поймать Штирлица с поличным, но тот всё время выкручивался. Как-то раз Штирлиц просматривал электронную почту. В это время незаметно вошел Мюллер и увидел, как у него на экране появился бессмысленный набор символов.

“Шифровка”, — подумал Мюллер.

“UTF-8”, — подумал Штирлиц.

Известно, что Штирлиц шифрует текст простым алгоритмом: он многократно вставляет в произвольное место текста две одинаковые буквы. Вы должны восстановить исходный текст. В исходном тексте нет двух одинаковых букв подряд.

Формат входного файла

В единственной строке записана шифровка Штирлица, состоящая из строчных латинских букв. Длина шифровки не превосходит 200000.

Формат выходного файла

Выведите восстановленный текст.

Примеры

decode.in	decode.out
wwstdaadierfflitzzz	stierlitz

Задача B. Проверка ПСП

Имя входного файла: `brackets.in`
Имя выходного файла: `brackets.out`

Дана строка, состоящая из круглых, квадратных и фигурных скобок. Нужно проверить, является ли она правильной скобочной последовательностью.

Формат входного файла

Во входном файле записана скобочная последовательность длиной не более 10000 символов.

Формат выходного файла

Выведите `YES`, если скобочная последовательность является правильной, и `NO` в противном случае.

Примеры

brackets.in	brackets.out
([]())	YES
([)]	NO

Задача C. Постфиксная запись

Имя входного файла: `postfix.in`
Имя выходного файла: `postfix.out`

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 входного файла

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

Формат выходного файла

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

Примеры

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

Задача D. Минимум на стеке

Имя входного файла: `stack-min.in`
Имя выходного файла: `stack-min.out`

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 входного файла

В первой строке входного файла задано одно целое число n — количество операций ($1 \le n \le 10^6$). 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 ($-10^9 \le x \le 10^9$). Гарантируется, что перед каждой операцией удаления или нахождения минимума структура не пуста.

Формат выходного файла

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

Примеры

stack-min.in	stack-min.out
8 1 2 1 3 1 -3 3 2 3 2 3	-3 2 2

Задача E. Строй новобранцев

Имя входного файла: `formation.in`
Имя выходного файла: `formation.out`

n новобранцев, пронумерованных от 1 до n, разделены на два множества: *строй* и *толпа*. Вначале *строй* состоит из новобранца номер 1, все остальные составляют *толпу*. В любой момент времени *строй* стоит в один ряд по прямой.

Товарищ сержант может использовать четыре команды. Вот они.

- «I, встать в строй слева от J.» Эта команда заставляет новобранца номер I, находящегося в *толпе*, встать слева от новобранца номер J, находящегося в *строю*.
- «I, встать в строй справа от J.» Эта команда действует аналогично предыдущей, за исключением того, что I встает справа от J.
- «I, выйти из строя.» Эта команда заставляет выйти из строя новобранца номер I. После этого он присоединяется к *толпе*.
- «I, назвать соседей.» Эта команда заставляет глубоко задуматься новобранца номер I, стоящего в *строю*, и назвать номера своих соседей по *строю*, сначала левого, потом правого. Если кто-то из них отсутствует (новобранец находится на краю ряда), то вместо соответствующего номера он должен назвать 0.

Известно, что ни в каком случае строй не остается пустым.

Иногда строй становится слишком большим, и товарищ сержант уже не может проверять сам, правильно ли отвечает новобранец. Поэтому он попросил вас написать программу, которая помогает ему в нелегком деле обучения молодежи и выдает правильные ответы для его команд.

Формат входного файла

В первой строке находятся два числа N ($1 \le N \le 75000$) и M ($1 \le M \le 75000$) — количество новобранцев и команд соответственно. Следующие M строк содержат команды, одна команда на строку.

Каждая команда — одна из следующих:

- `left I J` — соответствует команде «I, встать в строй слева от J.»
- `right I J` — «I, встать в строй справа от J.»
- `leave I` — «I, выйти из строя.»
- `name I` — «I, назвать соседей.»

Гарантируется, что все команды корректны, например, `leave I` не будет заставлять выйти из строя новобранца, стоящего в *толпе*. Также гарантируется, что *строй* никогда не будет пустым.

Формат выходного файла

Для каждой строки, содержащей `name I`, выведите в отдельной строке два числа — номера левого и правого соседа новобранца номер I. Если кто-то из соседей отсутствует, выведите ноль вместо его номера.

Пример

formation.in	formation.out
3 3 left 2 1 right 3 1 name 1	2 3
3 4 left 2 1 right 3 1 leave 1 name 2	0 3

Задача F. Очередь в поликлинике

Имя входного файла: `hospital.in`
Имя выходного файла: `hospital.out`

Очередь в поликлинике работает по сложным правилам. Обычные пациенты при посещении должны вставать в конец очереди. Пациенты, которым «только справку забрать», встают ровно в ее середину, причем при нечетной длине очереди они встают сразу за центром. Напишите программу, которая отслеживает порядок пациентов в очереди.

Формат входного файла

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

- «+ i» — к очереди присоединяется пациент i ($1 \leqslant i \leqslant N$) и встает в ее конец;
- «* i» — пациент i встает в середину очереди ($1 \leqslant i \leqslant N$);
- «-» — первый пациент в очереди заходит к врачу. Гарантируется, что на момент каждого такого запроса очередь будет не пуста.

Формат выходного файла

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

Примеры

hospital.in	hospital.out
7 + 1 + 2 - + 3 + 4 - -	1 2 3
10 + 1 + 2 * 3 - + 4 * 5 - - - -	1 3 2 5 4

Задача G. Парикмахерская

Имя входного файла: `saloon.in`
Имя выходного файла: `saloon.out`

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 входного файла

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 выходного файла

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 H. Гемоглобин

Имя входного файла: `hemoglobin.in`
Имя выходного файла: `hemoglobin.out`

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.

При этом 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 нём из базы.

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

Формат входного файла

В первой строке входного файла задано число n ($1 \le n \le 50\,000$) — количество обращений к базе данных. Запросы к базе выглядят следующим образом: «`+x`» ($1 \le x \le 10^9$) — добавить пациента с уровнем гемоглобина x в базу, «`-`» — удалить последнего пациента из базы, «`?k`» ($1 \le k \le 50\,000$) — вывести суммарный гемоглобин последних k пациентов. Гарантируется, что k не превосходит числа пациентов в базе. Также гарантируется, что запросов на удаление к пустой базе не поступает. Перед началом работы база данных пуста.

Формат выходного файла

Для каждого запроса «`-`» в отдельной строке выведите уровень гемоглобина в крови удаляемого пациента, а для каждого запроса «`?k`» — суммарный гемоглобин у последних k поступивших пациентов. Ответы выводите в порядке поступления запросов.

Примеры

hemoglobin.in	hemoglobin.out
7 +1 +2 +3 ?2 - - ?1	5 3 2 1

Задача I. Бюрократия

Имя входного файла: `bureaucracy.in`
Имя выходного файла: `bureaucracy.out`

В министерстве бюрократии одно окно для приема граждан. Утром в очередь встают n человек, i-й посетитель хочет получить a_i справок. За один прием можно получить только одну справку, поэтому если после приема посетителю нужны еще справки, он встает в конец очереди. За время приема министерство успевает выдать m справок. Остальным придется ждать следующего приемного дня.

Ваша задача — сказать, сколько еще справок хочет получить каждый из оставшихся в очереди посетитель в тот момент, когда прием закончится. Если все к этому моменту разойдутся, выведите -1.

Формат входного файла

В первой строке — количество посетителей n ($1 \leqslant n \leqslant 10^5$) и количество справок m ($0 \leqslant m \leqslant 10^9$). Во второй строке для каждого посетителя в порядке очереди указано количество справок a_i ($1 \leqslant a_i \leqslant 10^6$), которое он рассчитывает получить. Номером посетителя называется его место в исходной очереди.

Формат выходного файла

В первой строке выведите, сколько посетителей останется в очереди, когда прием закончится. Во второй строке выведите состояние очереди на тот момент, когда прием закончится: для всех посетителей по порядку выведите по одному числу через пробел — количество справок, которое он хочет еще получить. В случае, если в очереди никого не останется выведите одно число: -1

Примеры

bureaucracy.in	bureaucracy.out
3 2 1 2 3	2 3 1
4 5 2 5 2 3	3 4 1 2