Задача A. Стек на списке

Имя входного файла: `stack2.in`
Имя выходного файла: `stack2.out`

В этой задаче обязательно использовать самостоятельно написанный односвязный список.

Реализуйте работу стека. 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 стек числа N, по модулю не превышающего 10^9. Команда "`-`" означает изъятие элемента из стека. Гарантируется, что не происходит извлечения из пустого стека.

Формат входного файла

В первой строке входного файла содержится количество команд — M ($1 \leqslant M \leqslant 10^6$). Каждая последующая строка исходного файла содержит ровно одну команду.

Формат выходного файла

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

Пример

stack2.in	stack2.out
6 + 1 + 10 - + 2 + 1234 -	10 1234

Задача B. Правильная скобочная последовательность

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

Входной файл содержит несколько строк, каждая из которых содержит последовательность символов '`(`', '`)`', '`[`' и '`]`'. Выясните, является ли она правильной скобочной последовательностью с двумя типами скобок.

Используйте реализацию стека из задачи 1 или 2, на ваше усмотрение.

Пример

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

Пример

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

Задача D. Очередь на списке

Имя входного файла: `queue2.in`
Имя выходного файла: `queue2.out`

В этой задаче обязательно использовать самостоятельно написанный связный список.

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 входного файла

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

Формат выходного файла

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

Пример

queue2.in	queue2.out
4 + 1 + 10 - -	1 10

Задача E. Очередь с минимумом

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

В этой задаче обязательно использовать самостоятельно написанный вектор.

Реализуйте работу очереди. Необходимо отвечать на запрос о минимальном элементе, который сейчас находится в очереди. Для каждой операции изъятия элемента или запроса минимального элемента выведите ее результат.

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

Формат входного файла

В первой строке содержится количество команд — M ($1 \leqslant M \leqslant 500000$). В последующих строках содержатся команды, по одной в каждой строке.

Формат выходного файла

Для каждой операции поиска минимума в очереди выведите её результат. Гарантируется, что операций извлечения или поиска минимума для пустой очереди не производится.

Пример

queuemin.in	queuemin.out
7 + 1 ? + 10 ? - ? -	1 1 10

Задача F. Хип ли?

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

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 входного файла

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 выходного файла

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

Пример

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

Задача G. Приоритетная очередь

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

Реализуйте приоритетную очередь. Ваша очередь должна поддерживать следующие операции: добавить элемент, извлечь минимальный элемент, уменьшить элемент, добавленный во время одной из операций.

Все операции нумеруются по порядку, начиная с 1.

Формат входного файла

Входной файл содержит описание операций со очередью. В очередь помещаются и извлекаются только целые числа, не превышающие по модулю 10^9.

Формат выходного файла

Выведите последовательно результат выполнения всех операций `extract-min`. Если перед очередной операции `extract-min` очередь пуста, выведите вместо числа звездочку.

Пример

priorityqueue.in	priorityqueue.out
push 3 push 4 push 2 extract-min decrease-key 2 1 extract-min extract-min extract-min	2 1 3 *