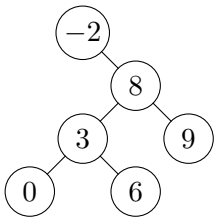
Задача A. Высота дерева (!) (1 балл)

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

Высотой дерева называется максимальное число вершин дерева в цепочке, начинающейся в корне дерева, заканчивающейся в одном из его листьев, и не содержащей никакую вершину дважды.

Так, высота дерева, состоящего из единственной вершины, равна единице. Высота пустого дерева (да, бывает и такое!) равна нулю. Высота дерева, изображенного на рисунке, равна четырем.



Дано двоичное дерево поиска. В вершинах этого дерева записаны ключи — целые числа, по абсолютному значению не превышающие 10^9. Для каждой вершины дерева V выполняется следующее условие:

- все ключи вершин из левого поддерева меньше ключа вершины V;
- все ключи вершин из правого поддерева больше ключа вершины V.

Найдите высоту данного дерева.

Формат входного файла

Входной файл содержит описание двоичного дерева. В первой строке файла находится число N ($0 \le N \le 200000$) — число вершин в дереве. В последующих N строках файла находятся описания вершин дерева. В $(i+1)$-ой строке файла ($1 \le i \le N$) находится описание i-ой вершины, состоящее из трех чисел K_i, L_i, R_i, разделенных пробелами — ключа в i-ой вершине ($|K_i| \le 10^9$), номера левого ребенка i-ой вершины ($i < L_i \le N$ или $L_i = 0$, если левого ребенка нет) и номера правого ребенка i-ой вершины ($i < R_i \le N$ или $R_i = 0$, если правого ребенка нет).

Все ключи различны. Гарантируется, что данное дерево является деревом поиска.

Формат выходного файла

Выведите одно целое число — высоту дерева.

Пример

height.in	height.out
6 -2 0 2 8 4 3 9 0 0 3 5 6 0 0 0 6 0 0	4

Примечание

Во входном файле задано то же дерево, что и изображено на рисунке.

Задача B. Проверка корректности (1 балл)

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

Свойство двоичного дерева поиска можно сформулировать следующим образом: для каждой вершины дерева V выполняется следующее условие:

- все ключи вершин из левого поддерева меньше ключа вершины V;
- все ключи вершин из правого поддерева больше ключа вершины V.

Дано двоичное дерево. Проверьте, выполняется ли для него свойство двоичного дерева поиска.

Формат входного файла

Входной файл содержит описание двоичного дерева. В первой строке файла находится число N ($0 \le N \le 200000$) — число вершин в дереве. В последующих N строках файла находятся описания вершин дерева. В $(i+1)$-ой строке файла ($1 \le i \le N$) находится описание i-ой вершины, состоящее из трех чисел K_i, L_i, R_i, разделенных пробелами — ключа в i-ой вершине ($|K_i| \le 10^9$), номера левого ребенка i-ой вершины ($i < L_i \le N$ или $L_i = 0$, если левого ребенка нет) и номера правого ребенка i-ой вершины ($i < R_i \le N$ или $R_i = 0$, если правого ребенка нет).

Формат выходного файла

Выведите «YES», если данное на входе дерево является двоичным деревом поиска, и «NO», если не является.

Примеры

check.in	check.out
6 -2 0 2 8 4 3 9 0 0 3 5 6 0 0 0 6 0 0	YES
0	YES
3 5 2 3 6 0 0 4 0 0	NO

Примечание

Решение данной задачи поможет Вам в дальнейшем проверять корректность построенных деревьев при отладке более сложных алгоритмов.

Задача C. Простое двоичное дерево поиска (2 балла)

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

Реализуйте двоичное дерево поиска.

Формат входного файла

Входной файл содержит описание операций с деревом, их количество не превышает 100. В каждой строке находится одна из следующих операций:

- `insert` x — добавить в дерево ключ x
- `delete` x — удалить из дерева ключ x. Если ключа x в дереве нет, то ничего делать не надо
- `exists` x — если ключ x есть в дереве выведите «true», если нет «false»
- `next` x — выведите минимальный элемент в дереве, строго больший x, или «none» если такого нет
- `prev` x — выведите максимальный элемент в дереве, строго меньший x, или «none» если такого нет

В дерево помещаются и извлекаются только целые числа, не превышающие по модулю 10^9.

Формат выходного файла

Выведите последовательно результат выполнения всех операций `exists`, `next`, `prev`. Следуйте формату выходного файла из примера.

Пример

bstsimple.in	bstsimple.out
insert 2 insert 5 insert 3 exists 2 exists 4 next 4 prev 4 delete 5 next 4 prev 4	true false 5 3 none 3

Задача D. Очередь с минимумом (1 балл)

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

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 E. Приоритетная очередь (2 балла)

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

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 единицы. Гарантируется, что размер очереди в процессе выполнения команд не превысит 10^6 элементов.

Формат входного файла

Входной файл содержит описание операций с 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 операцией `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 *

Задача F. Интерпретатор языка Quack (3 балла)

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

Язык Quack — забавный язык, который фигурирует в задаче G с Internet Problem Solving Contest 2004 (http://ipsc.ksp.sk/2004/real/problems/g.html). В этой лабораторной работе вам требуется написать интерпретатор языка Quack.

Виртуальная машина, на которой исполняется программа на языке Quack имеет внутри себя очередь, содержащую целые числа по модулю 65536 (`get` в описании операций означает извлечение из очереди, `put` — добавление в очередь). Кроме того, у виртуальной машины есть 26 регистров, которые обозначаются буквами от `a` до `z`.

`+`	Сложение: get x, get y, put (x+y) modulo 65536
`-`	Вычитание: get x, get y, put (x−y) modulo 65536.
`*`	Умножение: get x, get y, put (x*y) modulo 65536.
`/`	Целочисленное деление: get x, get y, put x div y. (будем считать, что 0 div 0 = 0)
`%`	Взятие по модулю: get x, get y, put x modulo y. (будем считать, что 0 mod 0 = 0)
`>`[register]	Положить в регистр: get x, установить значение [register] в x.
`<`[register]	Взять из регистра: put значение [register].
`P`	Напечатать: get x, вывести x в стандартный поток вывода и перевести строку.
`P`[register]	Вывести значение регистра [register] в стандартный поток вывода и перевести строку.
`C`	Вывести как символ: get x, вывести символ с ASCII кодом x modulo 256 в стандартный поток вывода.
`C`[register]	Вывести регистр как символ: вывести символ с ASCII кодом x modulo 256 (где x — значение регистра [register]) в стандартный поток вывода.
`:`[label]	Метка: эта строка программы имеет метку [label].
`J`[label]	Переход на строку с меткой [label].
`Z`[register][label]	Переход если 0: если значение регистра [register] равно нулю, выполнение программы продолжается с метки [label].
`E`[register1][register2][label]	Переход если равны: если значения регистров [register1] и [register2] равны, исполнение программы продолжается с метки [label].
`G`[register1][register2][label]	Переход если больше: если значение регистра [register1] больше, чем значение регистра [register2], исполнение программы продолжается с метки [label].
`Q`	Завершить работу программы. Работа также завершается, если выполнение доходит до конца программы.
[number]	Просто число во входном файле — put это число.

Формат входного файла

Входной файл содержит корректную синтаксически программу на языке Quack. Известно, что программа завершает работу не более чем за 100 000 шагов.

Формат выходного файла

Выведите содержимого стандартного потока вывода виртуальной машины в выходной файл.

Пример

quack.in	quack.out
100 0 :start >a Zaend <a <a 1 + - >b <b Jstart :end P	5050

Второй пример подразумевает UNIX-переводы строки (один символ с кодом 10).

quack.in	quack.out
58 49 10 62 97 10 80 97 10 90 97 50 10 60 97 10 74 49 10 58 50 10 48 10 58 51 10 62 97 10 90 97 52 10 67 97 10 74 51 10 58 52 10 0 :1 >a Pa Za2 <a J1 :2 0 :3 >a Za4 Ca J3 :4	58 49 10 62 97 10 80 97 10 90 97 50 10 60 97 10 74 49 10 58 50 10 48 10 58 51 10 62 97 10 90 97 52 10 67 97 10 74 51 10 58 52 10 0 :1 >a Pa Za2 <a J1 :2 0 :3 >a Za4 Ca J3 :4