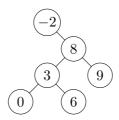
Задача А. Высота дерева (!) (1 балл)

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:

- ullet все ключи вершин из левого поддерева меньше ключа вершины V;
- ullet 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	4
-2 0 2	
8 4 3	
9 0 0	
3 5 6	
0 0 0	
6 0 0	

Примечание

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

Задача В. 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:

- \bullet все ключи вершин из левого поддерева меньше ключа вершины V;
- ullet 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	YES
-2 0 2	
8 4 3	
9 0 0	
3 5 6	
0 0 0	
6 0 0	
0	YES
3	NO
5 2 3	
6 0 0	
4 0 0	

Примечание

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

Задача С. 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:

- ullet insert x добавить в дерево ключ x
- ullet delete x- удалить из дерева ключ x. Если ключа x в дереве нет, то ничего делать не надо
- ullet exists x- если ключ x есть в дереве выведите «true», если нет «false»
- ullet next x выведите минимальный элемент в дереве, строго больший x, или «none» если такого нет
- $\bullet\,$ 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	true
insert 5	false
insert 3	5
exists 2	3
exists 4	none
next 4	3
prev 4	
delete 5	
next 4	
prev 4	

Задача D. 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 c 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 (то есть, числа от 0 до 65535, или беззнаковое 16-битное целое). Слово get в описании операций означает извлечение из очереди, put — добавление в очередь. Кроме того, у виртуальной машины есть 26 регистров, которые обозначаются буквами от а до z. Изначально все регистры хранят нулевое значение. В языке Quack существуют следующие команды:

нанды. +	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 $x \operatorname{div} 0 = 0$)
%	Взятие по модулю: get x, get y, put x modulo y. (будем
	считать, что $x \mod 0 = 0$)
>[register]	Положить в регистр: get x, установить значение [register]
	B 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.
С	Вывести как символ: get x, вывести символ с ASCII кодом
	х modulo 256 в стандартный поток вывода.
C[register]	Вывести регистр как символ: вывести символ с ASCII ко-
	дом х modulo 256 (где х — 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] рав-
	но 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 завершает-
	ся, 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	5050
0	
:start	
>a	
Zaend	
<a>	
<a>	
1	
+	
-	
>b	
 b	
Jstart	
:end	
P	

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

Лабораторная работа для групп M3305-M3307 Двоичное дерево поиска. Интерпретатор языка Quack. 2017 год.

SS	quack.in	quack.out
49		
10		
62		
97	62	62
10		
80 97 97 10 10 90 97 97 97 97 97 97 9		
97 10 90 97 97 97 550 560 10 60 60 60 97 10 10 10 10 10 10 10 10 10 10 10 10 10	80	
10	97	
90		
97	90	
50	97	
10 60 97 10 10 10 10 74 49 49 10 10 10 58 58 50 50 10 10 48 48 10 10 58 58 51 10 10 58 58 51 10 10 62 97 10 10 10 62 97 10 10 90 97 10 10 90 97 10 10 10 67 97 10 10 10 67 97 10 10 10 67 97 10 10 10 67 97 10 10 10 67 97 10 10 10 67 97 10 10 10 67 97 10 10 10 67 97 10 10 10 67 97 10 10 10 67 97 10 10 10 67 97 10 10 10 67 97 10 10 10 67 97 97 10 10 10 10 67 97 10 10 10 67 97 97 10 10 10 67 97 97 10 10 10 67 97 97 10 10 10 67 97 97 10 10 10 67 97 97 10 10 10 67 97 97 10 10 10 67 97 97 10 10 10 67 97 97 10 10 10 67 97 97 10 10 10 67 97 97 10 10 10 67 97 97 10 10 10 67 97 97 10 10 10 67 97 97 10 10 10 10 67 97 97 10 10 10 10 10 10 10 10 10 10 10 10 10		50
60 97 10 10 10 74 49 49 10 10 58 58 50 50 10 10 10 48 48 10 10 58 58 51 11 10 10 62 97 97 10 10 10 10 62 97 97 10 10 10 10 10 67 97 97 10 10 10 10 67 67 97 10 10 10 10 67 58 58 58 58 51 10 10 10 10 10 10 10 10 10 10 10 10 10		
97 10 74 49 49 49 10 10 58 58 50 50 10 10 48 48 10 10 58 58 51 10 62 97 10 10 62 97 97 10 10 10 90 90 97 52 52 52 10 10 10 10 10 10 10 67 97 97 10 10 10 10 10 10 58 58 58 51 11 10 10 00 07 11 10 10 00 00 11 10 10 00 00 11 10 10	60	
10 74 74 49 49 10 10 58 58 50 10 48 48 10 10 58 58 51 10 62 62 97 97 10 10 90 90 97 97 52 52 10 10 67 97 97 97 10 10 74 11 51 51 10 10 74 74 51 51 10 10 58 58 52 52 10 10 74 51 10 10 58 58 52 52 10 0 10 0 2 52 10 0 2 52 </td <td>97</td> <td></td>	97	
74 49 10 10 58 58 50 10 10 10 48 48 10 10 58 58 51 51 10 10 62 62 97 10 10 90 90 90 97 52 10 10 67 97 10 10 67 97 10 10 74 51 10 10 58 58 52 52 10 10 0 10 0 0 :1 :1 >a >a Pa Pa Za2 <a< td=""> 4a 2a 2a <a< td=""> 3 :3 >a >a Za4 <a< td=""> Ca <a< td=""> J3 J3</a<></a<></a<></a<>	10	
49 49 10 10 58 58 50 10 48 48 10 10 58 58 51 10 62 62 97 97 10 10 90 90 97 97 52 52 10 10 67 97 10 10 74 74 51 10 10 10 58 58 52 52 10 10 74 74 51 10 0 0 11 10 58 58 52 10 10 10 58 58 52 10 0 0 11 2a 2a2 2a2 4a 2a 2a 2a	74	74
10 10 58 58 50 10 48 48 10 10 58 58 51 10 62 62 97 10 10 10 90 90 97 97 52 52 10 10 67 97 10 10 67 97 10 10 74 74 51 10 58 58 52 52 10 10 58 58 52 52 10 10 68 58 52 52 10 10 74 51 10 10 10 10 20 0 11 11 2a 2a 4a 2a 31 3 2a 2a 4a 2a 3a 3 3a 3 3a 3 3a 3 3a <t< td=""><td>49</td><td>49</td></t<>	49	49
58 58 50 10 10 10 48 48 10 10 58 58 51 11 10 62 97 97 10 10 90 90 97 97 52 10 10 10 67 97 10 10 74 74 51 10 58 58 52 10 10 10 58 58 52 10 0 0 :1 >a Pa Pa Za2 <a< td=""> 4a J1 11 J1 :2 0 0 0 :3 >a 2a4 2a4 Ca Ca J3 J3</a<>	10	10
50 50 10 48 10 48 10 58 58 51 10 10 62 97 10 10 90 90 97 52 10 10 67 97 10 10 67 97 10 10 74 51 51 51 10 10 58 58 52 52 10 10 0 10 0 11 >a >a Pa Pa Za2 <a< td=""> <a< td=""> JJ 1:2 :2 0 :3 >a >a Za4 Ca Ca J3</a<></a<>	58	58
48 10 58 58 51 10 62 62 97 10 10 90 90 90 97 52 10 67 97 10 10 67 97 10 10 74 51 10 10 10 58 58 52 10 10 0 :1 >1 >a Pa Za2 <a< td=""> <a< td=""> J1 :2 0 :3 >a Za4 Ca Ca J3</a<></a<>	50	
10 58 58 51 10 10 62 97 10 10 90 90 97 52 10 10 67 97 10 67 97 10 74 51 51 10 58 58 52 52 10 10 58 58 52 52 10 0 :1 >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 0 :3 >a Za4 <a< td=""> Ca <a< td=""> J3 J3</a<></a<></a<></a<>		
58 58 51 10 62 62 97 97 10 90 90 90 97 97 52 52 10 67 97 10 74 74 51 10 58 58 52 10 0 10 58 58 52 10 0 0 :1 >a Pa 2a 2a 2a 4a J1 :2 0 0 :3 >a 2a4 Ca J3		
51 10 62 62 97 10 90 90 97 52 10 67 67 97 10 67 97 10 74 51 51 10 74 51 10 10 58 58 52 10 10 0 :1 >a Pa Pa Za2 <a< td=""> 4a J1 :2 0 :3 >a Za4 <a< td=""> Ca Ca J3 J3</a<></a<>	10	10
10 62 97 97 10 90 90 90 97 52 10 67 97 10 67 97 10 10 74 51 10 10 58 58 52 10 0 10 0 0 :1 >a Pa Pa Za2 <a< td=""> <a< td=""> <a< td=""> J1 :1 >a <a< td=""> J1 :2 0 :3 >a <a< td=""> Za4 <a< td=""> Ca <a< td=""> J3 J3</a<></a<></a<></a<></a<></a<></a<>		
62 97 10 90 90 97 52 10 67 97 10 67 97 10 10 67 97 10 10 74 51 10 58 58 58 52 10 0 0 10 0 0 11 >a Pa Za2		
97 10 90 90 97 52 10 67 97 68 97 10 67 97 10 10 74 51 10 58 51 10 58 52 10 10 58 52 10 0 10 58 52 10 10 58 52 2 2 2 2 2 2 3 4 2 4 3 J1 1 2 0 1 3 3 > a 2 4 Ca J3	10	10
10 90 97 52 10 67 97 10 67 97 10 74 51 10 58 52 10 0 10 58 58 52 10 0 0 :1 >a Pa Za2 <a 0="" :2="" :3="" <a="" ji="" pa="" sa="" za2="">a Za4 Ca Ji Ji 3 >a Za4 Ca Ji Ji	62	
90 97 52 10 67 97 10 67 97 10 67 97 10 10 74 51 10 58 52 10 10 58 58 52 10 0 0 :1 >a Pa Za2 <a 0="" :2="" :3="" <a="" ji="" ya="" za2="">a Za4 Ca Ji Ji 3 >a Za4 Ca Ji Ji		
97 52 10 67 97 97 10 67 97 10 10 74 51 10 10 58 58 58 52 10 10 0 10 0 :1 :1 >a Pa Za2 <a 0="" :2="" :3="" j1="">a Za4 Ca J3 Za4 Ca J3 J3 J3		
52 52 10 67 67 97 10 97 10 10 74 51 51 10 58 58 52 52 10 0 0 10 0 11 >a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 1:2 2 0 3 >a >a Za4 Ca Ca J3</a<></a<>	90	
10 67 67 67 67 97 97 10 10 74 74 74 51 10 10 58 58 52 52 52 52 10 0 0 11 20 20 20 20 20 20 20 20 20 20 20 20 20		
67 97 10 10 74 51 11 51 10 58 58 52 10 0 0 :1 :1 >a Pa Za2 <a 0="" 3="" :2="" :3="" j1="">a Za4 Ca J3 J3 J3		
97 10 74 74 51 51 10 58 58 52 10 0 0 :1 >a Pa Za2 <a 0="" :2="" :3="" j1="">a J1 :2 0 0 :3 >a Za4 Ca J3 J3 J3		
10 74 74 51 51 10 10 10 10 58 58 58 52 10 10 0 10 0 :1 >a >a Pa Pa Pa Za2 <a< td=""> <a< td=""> <a< td=""> J1 J1 :2 :2 0 :3 >a :3 >a Za4 Ca Ca J3 J3</a<></a<></a<>		
74 74 51 51 10 10 58 58 52 52 10 0 0 :1 >a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 :2 0 0 :3 >a Za4 Za4 Ca J3</a<></a<>		
51 10 58 58 52 52 10 10 0 11 >a >a Pa Pa Za2 Za2 <a< td=""> <a< td=""> J1 J1 :2 0 0 0 :3 >a Za4 Za4 Ca J3</a<></a<>		
10 58 52 52 10 10 0 0 :1 :1 >a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 :2 0 :3 >a >a Za4 Za4 Ca J3</a<></a<>		
58 52 10 10 0 0 :1 :1 >a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 <a< td=""> 0 :3 >a >a Za4 Ca J3 J3</a<></a<></a<>		
52 10 10 0 0 0 :1 >a >a Pa >a Pa Za2 <a< td=""> <a< td=""> <a< td=""> J1 J1 :2 0 0 :3 >a >a Za4 Ca J3 J3 J3 J3</a<></a<></a<>		
10 10 0 0 :1 :1 >a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 :2 0 :3 >a :3 >a Za4 Ca J3</a<></a<>		
0 :1 >a :1 >a Pa Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 :2 0 :3 >a :3 >a Za4 Ca J3 J3 J3</a<></a<>	10	
:1 >a >a Pa Pa Za2 <a< td=""> J1 :2 :2 0 :3 >a :3 >a Za4 Ca J3 J3 J3</a<>		
>a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 :2 0 0 :3 >a >a Za4 Ca J3 J3 J3</a<></a<>		
Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 :2 0 0 :3 :3 >a >a Za4 Ca Ca J3</a<></a<>		
Za2 <a< td=""> <a< td=""> J1 J1 :2 :2 0 0 :3 :3 >a >a Za4 Ca Ca J3</a<></a<>		
<a< td=""> <a< td=""> J1 :2 :2 0 :3 :3 >a >a Za4 Za4 Ca J3 J3 J3</a<></a<>		
J1 :2 0 :2 0 :3 >a :3 >a >a Za4 Za4 Ca J3		
:2		
0 :3 :3 :3 >a >a Za4 Ca		
:3 >a Za4 Ca J3	0	0
Za4 Za4 Ca Ca J3 J3	:3	:3
Ca Ca J3 J3		
J3 J3		
:4		
	:4	:4