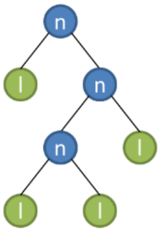
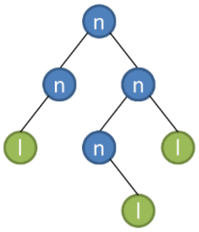
Task 1

Binary trees can sometimes be very difficult to work with. Fortunately, there is a class of trees with some really nice properties. A rooted binary tree is called “nice”, if every node is either a leaf, or has exactly two children.

For example, the following tree is nice,



but the following tree is not.



The leaves of a nice binary tree are labeled by the letter ‘l’, and other nodes are labeled by the letter ‘n’.

Given the pre-order traversal of a nice binary tree, you are required to find the depth of the tree.

Notes :

1. The depth of a tree is defined as the length of the longest path with one end at the root.

2. The pre-order traversal of the tree in the first image above produces the string “nlnnlll”.

Input

The first line contains the number of test cases T. T lines follow. Each line contains a string, which represents the pre-order traversal of a “nice” binary tree. Leaves are represented by the letter ‘l’ and other nodes by the letter ‘n’. The input is guaranteed to be the preorder traversal of a nice binary tree.

Output

Output one line for each test case, containing a single integer, the depth of tree.

Input	Output
7	0
l	2
nlnll	3
nlnnlll	3
nnnllll	1
nll	4
nlnlnlnll	4
nnlnllnlnlnll	

Task 2

Given a binary search tree (BST), which is represented in arrays as an implicit data structure of complete binary tree. In this structure, if a node has an index i, its children (if any) are found at indices 2i+1 and 2i+2, while its parent (if any) is found at index floor((i-1)/2). An AVL tree is a self-balancing BST where the Balance Factor (balanceFactor = height(left subtree) - height(right subtree)) of every node is -1, 0 or +1. Otherwise, it is not an AVL tree. Find whether the given BST is an AVL tree or not.

Input

The input begins with the number t of test cases in a single line (1 <= t <= 100). Each test case beings on a new line with a nonnegative integer n followed by n integers separated by spaces (0 <= n <= 100), where n is the number of places required to store the BST in the array representation. The array representation of the binary tree would have node values (0 <= node value <= 10000), and null-nodes are represented as -1s. The given binary tree is guaranteed to be a BST.

Output

For each test case, print T or F on a new line to indicate whether the given BST is an AVL tree or not, respectively.

Input	Output
12	T
0	T
1 10	T
2 20 10	T
3 20 10 30	F
4 30 20 -1 10	F
5 30 10 -1 -1 20	T
7 20 10 30 -1 -1 -1 40	F
7 10 -1 20 -1 -1 -1 30	T
7 40 20 60 10 30 50 70	F
10 40 20 60 -1 30 50 70 -1 -1 25	T
11 10 05 20 04 07 12 -1 02 -1 -1 08	F
11 10 05 20 04 07 -1 -1 02 -1 -1 08	