Size-based Tree Encoding and Recursive Traversals X30150_en

An old-fashioned encoding of binary trees, to be found in the complete works of the great Professor Donald Ervin Knuth, uses tree sizes instead of marking the empty trees at the leaves. Thus, after the size of the tree, the nodes with their information are provided, in preorder, and, together with each, we get the size of the left subtree. For instance, the encoding of the tree that can be seen by clicking on the pdf link of problem P98436 would be:
10 3 4 0 2 7 0 4 0 2 0 5 1 4 0 7 2 6 0 1 0
where 10 is the total size, 4 is the size of the left subtree of 3, 2 is the size of the left subtree of 0, and so on. If the first integer, the size, is n, then the whole encoding uses $2n + 1$ values. Write a program that reads a tree encoded thus and outputs its postorder and inorder.

Input

A size-encoded binary tree: first comes the size of the tree, then its preorder traversal where each node information is followed by the size of the corresponding left subtree. Empty trees are omitted altogether.

Output

The postorder and inorder traversals of the given tree, in two separate lines; make sure that there is a blank space after each element.

Sample input 1

```
10 3 4 0 2 7 0 4 0 2 0 5 1 4 0 7 2 6 0 1 0
```

Sample output 1

```
post: 4 7 2 0 4 1 6 7 5 3
 in: 7 4 0 2 3 4 5 6 1 7
```

Sample input 2

```
1 11 0
```

Sample output 2

```
post: 11
in: 11
```

Observation

In case you are using the `pytokr` module, be aware that you should not need the iterator: the function that reads the next word suffices.

Problem information

Author : José Luis Balcázar
Generation : 2024-02-06 11:48:59