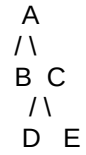
L2: Recursive programming in Lisp (2)

Write recursive Lisp functions for the following problems (optionally, you may use MAP functions):

A binary tree is memorised in the following two ways:

(node no-subtrees list-subtree-1 list-subtree-2 ...) (1)
(node (list-subtree-1) (list-subtree-2) ...) (2)

As an example, the tree



is represented as follows:

(A 2 B 0 C 2 D 0 E 0) (1)
(A (B) (C (D) (E))) (2)

Except for problems 4 and 9, conversion between types is not allowed; a direct method should be used instead.

1. For a given tree of type (1) return the path from the root node to a certain given node X.

2. Return the list of nodes on the k-th level of a tree of type (1).

3. Return the number of levels of a tree of type (1).

4. Convert a tree of type (2) to type (1).

5. Return the level (depth) of a node in a tree of type (1). The level of the root element is 0.

6. Return the list of nodes of a tree of type (1) accessed inorder.

7. Return the level of a node X in a tree of type (1). The level of the root element is 0.

8. Return the list of nodes of a tree of type (2) accessed inorder.

9. Convert a tree of type (1) to type (2).

10. Return the level of a node X in a tree of type (2). The level of the root element is 0.

11. Return the level (and coresponded list of nodes) with maximum number of nodes for a tree of type (2). The level of the root element is 0.

12. Determine the list of nodes accesed in preorder in a tree of type (2).

13. For a given tree of type (2) return the path from the root node to a certain given node X.

14. Determine the list of nodes accesed in postorder in a tree of type (1).

15. Determine the list of nodes accesed in postorder in a tree of type (2).

16. Determine if a tree of type (2) is ballanced (the difference between the depth of two subtrees is equal to 1).