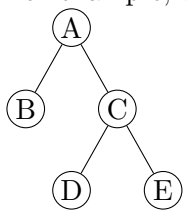
`L1: Recursive Programming in List (2)`

For the following problems it is required to write a recursive function (maybe using MAP functions).
A binary tree may be stored in one of two ways:

(1) `(node no-subtrees subtree-list1 subtree-list2 ...)`

(2) `(node (subtree-list1) (subtree-list2))`

For example, the following binary tree:



Can be represented in the two ways mentioned above respectively:

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

(2) `(A (B) (C (D) (E)))`

Except for problems 6 and 7, it is not allowed to use type conversions.

1. Given a tree of type (1). Display the path from the root to a given node x.
2. Display the list of nodes at level k from a tree of type (1).
3. Given a tree of type (1), Determine the height of the tree.
4. Convert a tree of type (2) into a tree of type (1).
5. Write a function that returns the depth of a node in a type (1) tree.
6. Return the list of nodes of a type (1) tree in in-order.
7. Given a type (1) tree, determine the level of a node x given that the root is at level 0.
8. Return the list of nodes of a type (2) tree in in-order.
9. Convert a tree of type (1) into a tree of type (2).
10. Given a type (2) tree, determine the level of a node x given that the root is at level 0.
11. Given a type (2) tree, determine the level with the most nodes and the associated list of nodes, considering the level of the root to be 0.
12. Return the list of nodes of a type (2) tree in pre-order.
13. Given a tree of type (2). Display the path from the root to a given node x.
14. Return the list of nodes of a type (2) tree in post-order.
15. Return the list of nodes of a type (1) tree in post-order.
16. Determine if a type (2) tree is ballanced, i.e., the difference in depth of the two subtrees is at most 1.