L3: MAP Functions in LISP programming

Solve the following problems using MAP functions.

1. Write a function to check if an atom is member of a list (the list is non-liniar)
2. Write a function that returns the sum of numeric atoms in a list, at any level.
3. Define a function to tests the membership of a node in a n-tree represented as (root list_of_nodes_subtree1 ... list_of_nodes_subtreen)
   Eg. tree is (a (b (c)) (d) (E (f))) and the node is "b" => true
4. Write a function that returns the product of numeric atoms in a list, at any level.
5. Write a function that computes the sum of even numbers and the decrease the sum of odd numbers, at any level of a list.
6. Write a function that returns the maximum of numeric atoms in a list, at any level.
7. Write a function that substitutes an element E with all elements of a list L1 at all levels of a given list L.
8. Write a function to determine the number of nodes on the level k from a n-tree represented as follows: (root list_nodes_subtree1 ... list_nodes_subtreen)
   Eg: tree is (a (b (c)) (d) (e (f))) and k=1 => 3 nodes
9. Write a function that removes all occurrences of an atom from any level of a list.
10. Define a function that replaces one node with another one in a n-tree represented as: root list_of_nodes_subtree1... list_of_nodes_subtreen)
    Eg: tree is (a (b (c)) (d) (e (f))) and node 'b' will be replace with node 'g' => tree (a (g (c)) (d) (e (f)))}
11. Write a function to determine the depth of a list.
12. Write a function that substitutes an element through another one at all levels of a given list.
13. Define a function that returns the depth of a tree represented as (root list_of_nodes_subtree1 ... list_of_nodes_subtreen)
    Eg. the depth of the tree (a (b (c)) (d) (e (f))) is 3
14. Write a function that returns the number of atoms in a list, at any level.
15. Write a function that reverses a list together with all its sublists elements, at any level.
16. Write a function that produces the linear list of all atoms of a given list, from all levels, and written in the same order. Eg.: (((A B) C) (D E)) --> (A B C D E)