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

```
(defun find-level (tree node &optional (depth 0))
  (when tree
    (let ((current (car tree))
          (num-children (cadr tree))
          (rest (cddr tree)))
      (cond
        ((eq current node) depth)

        ((zerop num-children)
         (find-level rest node depth))

        (t (or (find-level rest node (1+ depth))
               (find-level (skip-n rest num-children) node depth)))))))


(defun skip-n (lst n)
  (if (zerop n)
      lst
      (skip-n (skip-subtree lst) (1- n))))


(defun skip-subtree (tree)
  (if (zerop (cadr tree))
      (cddr tree)
      (skip-n (cddr tree) (cadr tree))))


(defun level (tree node)
  (find-level tree node 0))


(print (level '(A 2 B 0 C 2 D 0 E 0) 'A))
(print (level '(A 2 B 0 C 2 D 0 E 0) 'E))
```

(print (level '(A 2 B 0 C 2 D 0 E 0) 'B)) -> 1
(print (level '(A 2 B 0 C 2 D 0 E 0) 'E)) -> 2

Mathematical model:

Let T = (x, n, T_1, …, T_k)

level(x, T) = 0
level(node, T) = 1 + level(node, T_i) if node in subtree i

Otherwise:

level(node, T) undefined