Name: Galbenu Horia (ghie1544)

Group: 922

Lab code: L1

Lab delivery date: 19/12/2014

Problem number and statement: Problem 5

a) Write a function that returns an association list of two lists given as parameters.
b) Write a function to determine the number of all sublists of a given list, on any level.
c) Write the n-th element of a linear list twice.
d) Write a function to return the number of atoms in a list, at superficial level.

Formal descriptions:

- Mathematical models:

associate(l1,l2): { (), l1 null or l2 null
(…car(l1)), associate(cdr(l1), cdr(l2), otherwise

sublistnr(l): { 0, if l null
1+sublistnr(car(l))+sublistnr(cdr(l)), car(l) is a list
sublistnr(cdr(l)), otherwise

doublen(l, n): { (), l null
(…car(l),car(l),cdr(l)), n = 1
(…car(l),doublen(cdr(l),n-1)), otherwise

countatoms(l): { 0, l null
1 + countatoms(cdr(l)), car(l) is an atom
countatoms(cdr(l)), otherwise

- Meaning of function parameters:
  a) l1, l2 – input lists to create an association list
  b) l – input list of which we'll count the number of sublists
  c) l – input list, n – position of element to double
  d) l – input list of which we'll count the number of atoms

Source code:

```
( defun associate(l1 l2)

( cond ( (null l1) nil ) ( (null l2) nil )

(t ( append ( list ( cons ( car l1 ) ( car l2 ))) ( associate ( cdr l1 ) ( cdr l2 ))))))


( defun sublistnr (l)

  ( cond ( (null l) 0 ) ( (listp (car l)) (+ 1 ( sublistnr ( car l )) ( sublistnr ( cdr l ))) )

  (t ( sublistnr ( cdr l )))))


( defun doublen(l n)

  ( cond( (null l) nil ) ( (= n 1) ( append  ( list ( car l ) ( car l ))  ( cdr l )))

  (t ( cons ( car l ) ( doublen ( cdr l ) (- n 1))))


( defun countatoms(l)

  ( cond ( (null l) 0 ) (t ( +  ( cond

            ( ( atom ( car l ) ) 1 )

            ( t 0 )) ( countatoms ( cdr l ))))))
```

Running examples:

(associate ‘(a b c) ‘(1 2 3))->(append ((a.1)) (associate (b c) (2 3))->(append ((a.1)(b.2)) (associate (c) (3))->(append ((a.1)(b.2)(c.3)) (associate () ()) => ((a.1)(b.2)(c.3))

(sublist '(1 2 (3 (4 5)))->(+ 1 (sublistnr (1 2 (3 (4 5)))->(sublistnr (2(3(4 5)))->(sublistnr ((3(4 5)))->(+ 1 (sublistnr (3(4 5)) (sublistnr nil))->(sublistnr ((4 5)))->(+ 1 (sublistnr (4 5))… => 3

(doublen (1 2 3) 1)->(append (1 1) (2 3)) -> (1 1 2 3)

(countatoms (a b (c)))-> (+ 1 (countatoms b (c)) -> (+ 1 (countatoms (c)) -> (countatoms ()) => 2