Problem 5:

1. Write twice the n-th element of a linear list. Example: for (10 20 30 40 50) and n=3 will produce (10 20 30 30 40 50).

Code:

(defun twice (l n)

(cond

((null l) nil)

((= n 1) (cons (car l) l))

(T (cons (car l) (twice (cdr l) (- n 1))))

)

)

Function twice have two parameters : a list l and an integer n (the position of the element I want to write twice)

Condition 1: ((null l) nil) - Base case

If the list is empty -? Returns nil

Condition 2: ((= n 1) (cons (car l) l))

If n = 1 returns a new list constructs a new list where the first element of the original list l is duplicated at the beginning of the list. (car l - returns the first element of the list l; cdr (l) retrives the rest of the list after removing the first element; cons – creates a list)

Mathematical model:

twice(l1...ln,n) =

1. Write a function to return an association list with the two lists given as parameters.

Example: (A B C) (X Y Z) --> ((A.X) (B.Y) (C.Z)).

Code :

(defun my\_append (l k)

(if (null l)

k

(cons (car l) (my\_append (cdr l) k))

)

)

(defun association (l k)

(cond

((and (null l) (null k)) nil)

((null l) (my\_append (list (cons NIL (car k))) (association l (cdr k))))

((null k) (my\_append (list (cons (car l) NIL)) (association (cdr l) k)))

(T (my\_append (list (cons (car l) (car k))) (association (cdr l) (cdr k))))

)

)

My\_append( l, k ) – Has two parameters l and k, two lists

* if (null l) k - if list l is empty returns the second list as it is.
* (cons (car l) (my\_append (cdr l) k)) - If l is not empty, it takes the first element of l and cons (create a new pair) it with the result of a recursive call to my\_append on the rest of the list (cdr l) and the second list k. This process continues until the base case where l is empty.

Association (l,k) - Two parameters l and k, lists, and does the association of the elements

* (and (null l) (null k)) nil) - If both l and k are empty, it returns nil.
* ((null l) (my\_append (list (cons NIL (car k))) (association l (cdr k)))) - If l is empty, it pairs each element of k with NIL and appends the result
* ((null k) (my\_append (list (cons (car l) NIL)) (association (cdr l) k))) - If k is empty, it pairs each element of l with NIL and appends the result
* (T (my\_append (list (cons (car l) (car k))) (association (cdr l) (cdr k)))) - Otherwise, it pairs the first elements of both lists and appends the result

Mathematical model:

My\_append(l1..ln ,k1..km ) =

Association(l1..ln, k1..km ) =

1. Write a function to determine the number of all sublists of a given list, on any level. A sublist is either the list itself, or any element that is a list, at any level. Example: (1 2 (3 (4 5) (6 7)) 8 (9 10)) => 5 lists: (list itself, (3 ...), (4 5), (6 7), (9 10)).

Code :

(defun countLists(l)

(cond

((null l) 1)

((listp (car l)) (+ (countLists (car l)) (countLists (cdr l))))

(t (countLists (cdr l)))

)

)

The function countLists( l ) – has one parameter l, a list, in which we search the sublists

* ((atom l) 0) - if the list is actually an element returns 0
* ((null l) 1) - If l is null (empty list), it returns 1, indicating the base case
* ((listp (car l)) (+ (countLists (car l)) (countLists (cdr l)))) - If the first element of l is a list, it adds the count of sublists
* (t (countLists (cdr l))) - If the first element of l is not a list, it recursively calls countLists on the rest of the list

Mathematical Model:

countLists(l1..ln) =

d) Write a function to return the number of all numerical atoms in a list at superficial level.

(defun countNumbers (l)

(cond

((null l) 0)

((numberp (car l)) (+ 1 (countNumbers (cdr l))))

(T (countNumbers (cdr l)))

)

)

Code:

The function countNumbers(l) have one parameter, l, which is a list. In this list we will count all the numerical atoms.

* ((null l) 0) - If l is null (empty list), it returns 0
* ((numberp (car l)) (+ 1 (countNumbers (cdr l)))) - If the first element of l is a number, it adds 1 to the count and recursively calls countNumbers on the rest of the list (cdr l)
* (T (countNumbers (cdr l))) - If the first element of l is not a number, it recursively calls countNumbers on the rest of the list

Mathematical model:

countNumbers(l) =