A

insertN(𝑙1𝑙2𝑙3…𝑙𝑛,𝑒, n)= 𝑒 ∪ 𝑙1𝑙2𝑙3…𝑙𝑛, 𝑖𝑓 n=1

𝑙1∪ insertN (𝑙2𝑙3…𝑙𝑛, 𝑒, n=n−1), 𝑜𝑡ℎ𝑒𝑟𝑤𝑖𝑠𝑒

B

suma(𝑙1𝑙2𝑙3…𝑙𝑛)= 0, 𝑖𝑓 n=0

𝑙1 + suma(𝑙2𝑙3…𝑙𝑛), if l1 is nr

suma(l1) + suma(l2l3..ln), if l1 is list

suma (l2l3…ln), otherwise

C

My\_append(l1l2..ln, a1a2..am) = a1a2..am, if n = 0

L1U my\_append(l2l3..ln, a1a2..am), altfel

Sub(𝑙1𝑙2𝑙3…𝑙𝑛)= [], 𝑖𝑓 n=0

l1 U my\_append (my\_append(sub(l1), l1), sub(l2l3..ln)), if l1 is list

Sub (l2l3…ln), otherwise

D

Rm (l1l2..ln, e) = [], if n = 0

Rm(l2l3..ln, e), l1 = e

L1 U Rm(l2l3..ln, e), l1 <> e

Eqq(l1l2..ln, a1a2..am) = true, n =0 and m = 0

False, n = 0 or m =0

Eqq(l1l2…ln, rm(a1a2..am, l1)), otherwise

;a

(defun insertN (l e n)

(cond

((null l) (cons e l))

((equal n 1) (cons e l))

(T (cons (car l) (insertN (cdr l) e (- n 1))))

)

)

;b

(defun suma(l)

(cond

((null l) 0)

((numberp (car l)) (+ (car l) (suma(cdr l))))

((listp (car l)) (+ (suma(car l)) (suma(cdr l))))

(T (suma(cdr l)))

)

)

;c

(defun my\_append (l1 l2)

(cond

((null l1) l2)

(T (cons (car l1) (my\_append (cdr l1) l2)))

)

)

(defun sub(l)

(cond

((null l) nil)

;((listp (car l)) (append (list (car l)) (sub (car l)) (sub (cdr l))))

((listp (car l)) ( my\_append (my\_append (sub (car l)) (list(car l)) ) (sub (cdr l )) ))

(T (sub(cdr l)))

)

)

;d

(defun rm (l e)

(cond

((null l) nil)

((equal (car l) e) (rm (cdr l) e))

(T (cons (car l) (rm (cdr l) e)))

)

)

(defun eqq(l1 l2)

(cond

((and (null l1) (null l2)) T )

((or (null l1) (null l2)) nil)

(T (eqq (cdr l1) (rm l2 (car l1))))

)

)