EXERCITIUL 1

(defun anexa (LL)

(do ((lista LL (cddr lista)) (final '() ))

((null lista) final)

(setq final (cons (car lista) final))

))

(anexa '(1 (2) (3 4) (5) 6 7 8))

(8 6 (3 4) 1)

(anexa '(1 2 3 4 5))

(5 3 1)

(defun anexa2 (L)

(if (null L)

nil

(if (atom (car L))

(cons (car L) (anexa2 (cdr L)))

(append (anexa2 (car L)) (anexa2 (cdr L))))))

(anexa2 (anexa '(1 (2) (3 4) (5) 6 7 8)))

(8 6 3 4 1)

(defun f (L) (apply '+ (anexa2 (anexa L))))

(f '(1 (2) (3 4) (5) 6 7 8))

22

(f '(1 (2) (3 4 1 1) (5) 6 7 8))

24

(defun anexa2 (L)

(if (null L) nil

(if (atom (car L))

(cons (car L) (anexa2 (cdr L)))

(append (anexa2 (car L)) (anexa2 (cdr L))))))

(defun f (L) (apply '+ (anexa2 (

(lambda (LL)

(do ((lista LL (cddr lista)) (final '() ))

((null lista) final)

(setq final (cons (car lista) final))

)) L)

)))

(f '(1 (2) (3 4 1 1) (5) 6 7 8))

24

EXERCITIUL 2

(defun numar\_aparitii (lista n)

(apply '+ (mapcar (lambda (x) (if (equal x n) 1 0)) lista)))

(defun anexa22 (L)

(do ((lista L (cdr lista)) (aux '()))

((null lista) aux)

(if (atom (car lista)) (setq aux (cons (car lista) aux)) (

do((idk (car lista) (car idk)))

((not(listp (car idk))) (setq aux (append aux idk)))

))))

(defun numar\_aparitii (lista n)

(apply '+ (mapcar (lambda (x) (if (equal x n) 1 0)) ((lambda (L)

(do ((lista L (cdr lista)) (aux '()))

((null lista) aux)

(if (atom (car lista)) (setq aux (cons (car lista) aux)) (

do((idk (car lista) (car idk)))

((not(listp (car idk))) (setq aux (append aux idk)))

)))

) lista ))))

(numar\_aparitii '(1 (2) (3 4 1 1) (5) 6 7 8) '1)

3