;PROBLEMA 1 - PROBLEMA 13

(defun apartine\_arbore(l elem)

(cond

((null l) nil)

((listp (car l)) (OR (apartine\_arbore (car l) elem) (apartine\_arbore (cdr l) elem) ) )

((equal (car l) elem) t)

(t (apartine\_arbore (cdr l) elem))

)

)

(defun path(l elem)

(cond

((null l) nil)

((equal (car l) elem) (list (car l)) )

((apartine\_arbore (cadr l) elem) (cons (car l) (path (cadr l) elem)))

((apartine\_arbore (caddr l) elem) (cons (car l) (path (caddr l) elem) ))

(t NIL)

)

)

;(path '(A (B) (C (D) (E (F) (G (H) (I))))) 'I)

;PROBLEMA 2

(defun nivel\_k (l k)

(cond

((null l) NIL)

((= k 1) (list (car l)))

(t (append (nivel\_k (cadr l) (- k 1)) (nivel\_k (caddr l) (- k 1)) ) )

)

)

;PROBLEMA 3

(defun nr\_niveluri(l)

(cond

((null l) 0)

((null (cdr l)) 1)

(t (+ 1 (max (nr\_niveluri (cadr l)) (nr\_niveluri (caddr l)))))

)

)

;PROBLEMA 5 - PROBLEMA 10

(defun apartine\_arbore(l elem)

(cond

((null l) nil)

((listp (car l)) (OR (apartine\_arbore (car l) elem) (apartine\_arbore (cdr l) elem) ) )

((equal (car l) elem) t)

(t (apartine\_arbore (cdr l) elem))

)

)

(defun nivel\_nod(l elem)

(cond

((null l) NIL)

((equal (car l) elem) 0)

((apartine\_arbore (cadr l) elem) (+ 1 (nivel\_nod (cadr l) elem)))

((apartine\_arbore (caddr l) elem) (+ 1 (nivel\_nod (caddr l) elem)))

(t NIL)

)

)

;PROBLEMA 6 - PROBLEMA 8

;SRD

(defun inordine(arb)

(cond

; frunza

((null arb) nil)

((null (cdr arb))

(list (car arb))

)

(t

(append

(inordine (cadr arb))

(list (car arb))

(inordine (caddr arb))

)

)

)

)

;PROBLEMA 11

;nodurile de pe nivelul k

(defun nivel\_k (l k)

(cond

((null l) NIL)

((= k 1) (list (car l)))

(t (append (nivel\_k (cadr l) (- k 1)) (nivel\_k (caddr l) (- k 1)) ) )

)

)

(defun lungime(l)

(cond

((null l) 0)

(t (+ 1 (lungime (cdr l))))

)

)

;incep cu i de la 0

(defun nivel\_maxim(l i)

(cond

;daca am trecut deja de nivelul maxim

((= (lungime (nivel\_k l i)) 0) 0)

((> (lungime (nivel\_k l i)) (lungime (nivel\_k l (nivel\_maxim l (+ i 1))))) i)

(t (nivel\_maxim l (+ i 1)))

)

)

(defun noduri\_nivel\_maxim(l)

(cons (list (nivel\_maxim l 0)) (list (nivel\_k l (nivel\_maxim l 0))))

)

;PROBLEMA 12

;RSD

(defun preordine(arb)

(cond

; frunza

((null arb) nil)

((null (cdr arb))

(list (car arb))

)

(t

(append

(list (car arb))

(preordine (cadr arb))

(preordine (caddr arb))

)

)

)

)

;PROBLEMA 14

;SDR

(defun postordine(arb)

(cond

; frunza

((null arb) nil)

((null (cdr arb))

(list (car arb))

)

(t

(append

(postordine (cadr arb))

(postordine (caddr arb))

(list (car arb))

)

)

)

)

;PROBLEMA 16

(defun nr\_niveluri(l)

(cond

((null l) 0)

((null (cdr l)) 1)

(t (+ 1 (max (nr\_niveluri (cadr l)) (nr\_niveluri (caddr l)))))

)

)

(defun echilibrat(l)

(cond

((null l) t)

(t (AND (< (abs (- (nr\_niveluri (cadr l)) (nr\_niveluri (caddr l)))) 2) (echilibrat (cadr l)) (echilibrat (caddr l))) )

)

)