# CURS 11

## Exemple funcții MAP (cont).

**EXEMPLU 2.1** Fie următoarele definiții

```
(defun f (L e)
        (list e L)
)

(setq L '(1 2 3))
(setq e 4)
```

(mapcar #'*f* L e) se evaluează la NIL
(mapcar #'(lambda (L) (*f* L e)) L) se evaluează la ((4 1) (4 2) (4 3))

**EXEMPLU 2.2** Fie următoarea definiție de funcție

```
(defun f (L)
        (list L)
)
```

(mapcar #'*f* '(1 2 3)) se evaluează la ((1) (2) (3))
(mapcan #'*f* '(1 2 3)) se evaluează la (1 2 3)

De remarcat echivalența între

(mapcan #'*f* L) și (apply #'append (mapcar #'*f* L))

**EXEMPLU 2.3** Să se definească o funcţie care să returneze lungimea unei liste neliniare (în numă de atomi la orice nivel)

(*LG* '(1 (2 (a ) c d) (3))) = 6

$$LG(L) = \begin{cases} 1 & daca \quad L\ e\ atom \\ \sum_{i=1}^{n} LG(L_i) & daca\ L\ e\ lista\ (L_1 \dots L_n) \end{cases}$$

```
(DEFUN LG (L)
        (COND
                ((ATOM L) 1)
                (T (APPLY #'+ (MAPCAR #'LG L)))
        )
)
```

**EXEMPLU 2.4** Să se definească o funcţie care având ca parametru o listă neliniară să returneze numărul de subliste (inclusiv lista) având lungime număr par (la nivel superficial).

(*nr* '(1 (2 (3 (4 5) 6)) (7 (8 9)))) = 4

Vom folosi o funcţie auxiliară care returnează T dacă lista argument are număr par de elemente la nivel superficial, NIL în caz contrar.

$$nr(L) = \begin{cases} 0 & daca\ L\ e\ atom \\ 1 + \sum_{i=1}^{n} lg(L_i) & daca\ L\ e\ lista\ (L_1 \dots L_n)\ si\ n\ e\ par \\ \sum_{i=1}^{n} lg(L_i) & altfel \end{cases}$$

```
(DEFUN PAR (L)
        (COND
                ((= 0 (MOD (LENGTH L) 2)) T)
                (T NIL)
        )
)

(DEFUN nr (L)
        (COND
                ((ATOM L) 0)
                ((PAR L) (+ 1 (APPLY #'+ (MAPCAR #'nr L))))
                (T (APPLY #'+ (MAPCAR #'nr L)))
        )
)
```

**EXEMPLU 2.5** Să se definească o funcţie care având ca parametru o listă neliniară să returneze lista atomilor (de la orice nivel) din listă.

(*atomi* '(1 (2 (3 (4 5) 6)) (7 (8 9)))) = (1 2 3 4 5 6 7 8 9)

$$atomi(L) = \begin{cases} (L) & daca\quad L\ e\ atom \\ \bigcup_{i=1}^{n} atomi(L_i) & daca\ L\ e\ lista\ (L_1 \dots L_n) \end{cases}$$

```
(DEFUN atomi (L)
        (COND
                ((ATOM L) (LIST L))
                (T (MAPCAN #'atomi L))
        )
)
```

**Observaţie:** Aceeaşi cerinţă ar putea fi rezolvată folosind funcţia MAPCAR

```
(DEFUN atomi (L)
        (COND
                ((ATOM L) (LIST L))
                (T (APPLY #'APPEND (MAPCAR #'atomi L)))
        )
)
```

**EXEMPLU 2.6** Să se definească o funcţie care determină numărul de apariții, de la orice nivel, ale unui element într-o listă neliniară.

(*nrap* 'a '(1 (a (3 (4 a) a)) (7 (a 9)))) = 4

$$nrap(e,l) = \begin{cases} 1 & daca\ l = e \\ 0 & dacă\ l\ e\ atom \\ \sum_{i=1}^{n} nrap(e,l_i) & altfel, l = (l_1 l_2 \dots l_n)\ e\ lista \end{cases}$$

```
(defun nrap(e L)
        (cond
                ((equal L e) 1)
                ((atom L) 0)
                (t (apply #'+ (mapcar #'(lambda(L)
                                                (nrap e L)
                                        )
                                        l
                                )
                   )
                )
        )
)
```

**EXEMPLU 2.7** Se dă o listă neliniară. Se cere să se returneze lista din care au fost șterși atomii numerici negativi. Se va folosi o funcție MAP.

Ex: (*stergere* '(a 2 (b -4 (c -6)) -1)) → (a 2 (b (c)))

$$sterg(l) = \begin{cases} \emptyset & daca\ l\ numeric\ negativ \\ l & dacă\ l\ e\ atom \\ sterg(l_i) & altfel, l = (l_1 l_2 \dots l_n)\ e\ lista \end{cases}$$

```
(defun sterg(L)
  (cond
    ((and (numberp L) (minusp L)) nil)
    ((atom L) (list L))
    (t (list (apply #'append                          ; (sterg ' ((a) (c -2))) → ((a) (c))
                    (mapcar #'sterg L)                ; (nconc '(a) '(c)) → (a c)
             )                                        ; (nconc '((a)) '((c))) → ((a) (c))

       )
     )
   )
 )

(defun stergere(L)
  (car (sterg L))
)
```

**EXEMPLU 2.8** Se dă un arbore n-ar nevid, reprezentat sub forma unei liste neliniare de forma (rădăcina lista_sub1.....lista_sub_n) (Varianta 2, Curs 9). Se cere să se determine numărul de noduri din arbore.

(*nrNoduri* '(a (b (c) (d (e))) (f (g)))) va produce 7

Model recursiv

$$nrNoduri(e, l_1 l_2 \dots l_n) = \begin{cases} 1 & daca\ n = 1 \\ \sum_{i=2}^{n} nrNoduri(e, l_i) & altfel \end{cases}$$

```
(defun nrNoduri(L)
        (cond
                ((null (cdr L)) 1)
                (t (+ 1 (apply #'+
                                (mapcar #'nrNoduri (cdr L))
                        )
                   )
                )
        )
)
```

**EXEMPLU 2.9** Se dă un arbore n-ar nevid, reprezentat sub forma unei liste neliniare de forma (rădăcina lista_sub1.....lista_sub_n) (Varianta 2, Curs 9). Se cere să se determine adâncimea arborelui. Observație. Nivelul rădăcinii este 0.

(*adancime* '(a (b (c) (d (e))) (f (g)))) va produce 3

Model recursiv

$$adancime(e, l_1 l_2 \dots l_n) = \begin{cases} 0 & daca\ n = 1 \\ \max\left(adancime(l_2), adancime(l_3), \dots, adancime(l_n)\right) & altfel \end{cases}$$

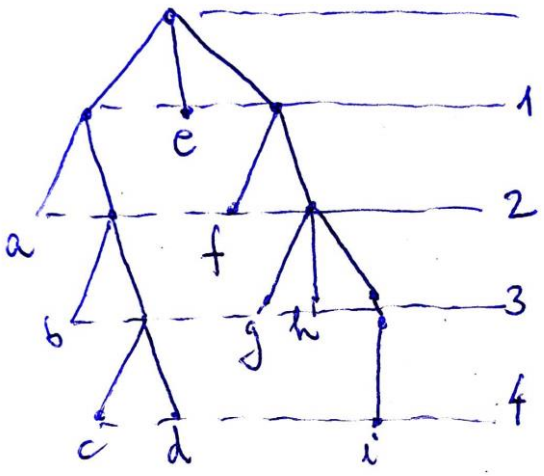
```
(defun adancime(L)
        (cond
                ((null (cdr L)) 0)
                (t (+ 1 (apply #'max
                                (mapcar #'adancime (cdr L))
                        )
                   )
                )
        )
)
```

**EXEMPLU 2.10** Să se determine lista atomilor de adâncime *n* dintr-o listă neliniară (nivelul superficial al listei se consideră 1).

(lista '((a (b (c d))) e (f (g h (i)))) 3) va produce (b g h)

(lista '((a (b (c d))) e (f (g h (i)))) 4) va produce (c d i)

(lista '((a (b (c d))) e (f (g h (i)))) 5) va produce NIL



Model recursiv

$$lista(l,n) = \begin{cases} (l) & dacă\ n = 0\ si\ l\ atom \\ \emptyset & dacă\ n = 0 \\ \emptyset & dacă\ l\ atom \\ \bigcup_{i=2}^{k} lista\ (l_i, n-1) & altfel, l = (l_1 l_2 \dots l_k)\ e\ lista \end{cases}$$

```
(defun lista(L n)
        (cond
                ((and (= n 0) (atom L)) (list L))
                ((= n 0) nil)
                ((atom L) nil)
                (t (mapcan #'(lambda(L)
                                (lista L (- n 1))
                             )
                           L
                   )
                )
        )
)
```

**EXEMPLU 2.11** Se dă o listă de liste. Se cere să se determine ....

```
        (defun m (L)
            (cond
                ((numberp L) L)
                ((atom L) most-negative-fixnum)
                (t (apply #'max
                          (mapcar #'m L)
                   )
                )
             )
        )

        (defun lista (L)
            (mapcan #'((lambda (v)
                                (cond
                                        ((= 0 (mod v 2) (list v))
                                        (t nil)
                                )
                       ) (m L)
                      )
            L)
        )
```

(*lista* '((5 a (2 b (8))) (7 a (9)) (c d (10)))) → (8 10)

**EXEMPLU 2.12** Se dă o listă liniară. Se cere....

```
(defun p (L)
   (mapcan #'(lambda (e1)
                     (mapcar #'(lambda (e2)
                                       (list e1 e2)
                               )
                       L
                      )
                )
               L
   )
)
```

(*p* '(1 2 3)) → ((1 1) (1 2) (1 3) (2 1) (2 2) (2 3) (3 1) (3 2) (3 3))

**EXEMPLU 2.13** O matrice se poate reprezinta sub forma unei liste formate din listele conținând elementele de pe linii, în ordine de la prima la ultima linie. De exemplu, lista ((1 2) (3 4)) corespunde matricei

1 2
3 4

Fie L o listă asociată unei matrici cu elemente numerice. Care este rezultatul returnat de funcția *fct*?

```
(defun fct (L)
    (cond
        ((null (car L)) nil)
        (t (cons
                (mapcar #'car L)
                (fct (mapcar #'cdr L))
           )
        )
     )
)
```

(*fct* '((1 2 ) (4 5) (7 8))) → ((1 4 7) (2 5 8))

**EXEMPLU 2.14** Să se definească o funcţie care având ca parametru o listă neliniară returnează lista liniară a atomilor care apar pe orice nivel, dar în ordine inversă.

(INVERS '(A (B C (D (E))) (F G))) = (G F E D C B A)

```
(DEFUN INVERS (L)
        (COND
                ((ATOM L) (LIST L))
                (T (MAPCAN #'INVERS (REVERSE L)))
        )
)
```

**EXEMPLU 2.15** O matrice se poate reprezenta în Lisp sub forma unei liste ale cărei elemente sunt liste reprezentând liniile matricei.

( (linia1) (linia2)….)

Să se definească o funcţie care având ca parametri două matrice de ordin **n** returnează (sub formă de matrice) produsul acestora.

(PRODUS '((1 2) (3 4)) '((2 -1) (3 1))) = ((8 1) (18 1))

**Observaţie:** Vom folosi două funcţii ajutătoare: o funcţie (COLOANE L) care returnează lista coloanelor matricei parametru L şi o funcţie (PR L1 L2) care returnează sub formă de matrice rezultatul înmulţirii matricei L1 (listă de linii) cu lista L2 (o listă de coloane ale unei matrice).

```
(DEFUN COLOANE (L)
        (COND
                ((NULL (CAR L)) NIL)
                (T (CONS (MAPCAR #'CAR L) (COLOANE (MAPCAR #'CDR L))))
        )
)


(DEFUN PR (L1 L2)
        (COND
                ((NULL (CAR L1)) NIL)
                (T (CONS (MAPCAR #'(LAMBDA (L)
                                        (APPLY #'+ (MAPCAR #'* (CAR L1) L))
                                   )
                                L2
                         )
                         (PR (CDR L1) L2)
                 )
                )
        )
 )


(DEFUN PRODUS (L1 L2)
        (PR L1 (COLOANE L2))
)
```

**EXEMPLU 2.16** Se dă o mulțime reprezentată sub forma unei liste liniare. Se cere să se genereze lista submulțimilor mulțimii. Se va folosi o funcție MAP.

Ex: (*subm* '(1 2)) → (nil (1) (2) (1 2))

Observație
(setq e 1)
(mapcar #' lambda(L) (cons e L)) '(2 3)) va produce ((1 2) (1 3))

```
(defun subm (L)
   (cond
        ((null L) (list nil))
        (t ((lambda (s)
                (append s (mapcar #'(lambda (sb)
                                        (cons (car L) sb)
                                    )
                                s
                                )
                )
             )
             (subm (cdr L)
             )
        )
    )
 )
```

**EXEMPLU 2.17** Se dă o mulțime reprezentată sub forma unei liste liniare. Se cere să se genereze lista permutărilor mu lțimii. Se va folosi o funcție MAP.

Ex: (*permutari* '(1 2 3)) → ((1 2 3) (1 3 2) (2 1 3) (2 3 1) (3 1 2) (3 2 1))

```
(defun permutari (L)
    (cond
        ((null (cdr L)) (list L))
        (t (mapcan #'(lambda (e)
                        (mapcar #'(lambda (p)
                                    (cons e p)
                                  )
                                  (permutari (remove e L))
                        )
                    )
                L
            )
        )
    )
)
```