```
;D. Se consideră o listă neliniară. Să se scrie o funcţie LISP care să aibă ca rezultat lista iniţială în care toate apariţiile unui
;element e au fost înlocuite cu o valoare e1. Se va folosi o funcţie MAP.
;Exemplu a) dacă lista este (1 (2 A (3 A)) (A)) e este A şi e1 este B => (1 (2 B (3 B)) (B))
; b) dacă lista este (1 (2 (3))) şi e este A => (1 (2 (3)))


; model matematic
; inlocuire(l, e, subs) = { l , l e atom si l != e
; { subs , l e atom si l = e
; { inlocuire(l1) U ... U inlocuire(l1) , altfel
;
; inlocuire(l:list, e:element, subs:element)
(defun inlocuire(l e subs)
    (cond
        ((AND (atom l) (not(equal l e))) l)
        ((AND (atom l) (equal l e)) subs)
        (t (mapcar #'(lambda (x) (inlocuire x e subs)) l))
    )
)

(defun main(l e subs)
    (inlocuire l e subs)
)
(print (main '(1 (2 A (3 A)) (A)) 'A 'B))

; Să se substituie un element  e prin altul e1 la orice nivel impar al unei liste neliniare. Nivelul superficial se consideră 1. De
; %exemplu, pentru lista (1 d (2 d (d))), e=d şi e1=f rezultă lista (1 f (2 d (f))).


; Model matematic:
; nivelImp(l, e, e1, niv) =
;     { e1                              , l atom si l = e si niv % 2 = 1
;     { l                              , l atom si l = e si niv % 2 = 0
;     { l                              , l atom
;     { nivelImp(l1,e,e1,niv+1) U ... U nivelImp(ln,e,e1,niv+1) , altfel
;
; Parametri:
; l: lista neliniara initiala
; e: elementul care trebuie inlocuit
; e1: elementul cu care se inlocuieste
; niv: nivelul curent in lista
(defun nivelImp(l e e1 niv)
    (cond
        ((and (atom l) (equal l e) (equal (mod niv 2) 1)) e1)
        ((and (atom l) (equal l e) (equal (mod niv 2) 0)) l)
        ((atom l) l)
        (t (mapcar #'(lambda (x) (nivelImp x e e1 (+ niv 1)))l))
    )
)
(defun main1(L e e1)
    (nivelImp L e e1 0)
)
(print (main1 '(1 d (2 d (d))) 'd 'f))
```

```
;D. Un arbore n-ar se reprezintă în LISP astfel (nod subarbore1 subarbore2 .....). Se cere să se verifice dacă
un nod  x apare pe
;un nivel par în arbore. Nivelul rădăcinii se consideră a fi 0. Se va folosi o funcție MAP.
;Exemplu pentru arborele (a (b (g)) (c (d (e)) (f)))
;a) x=g => T b) x=h => NIL


; Model matematic:
; adev(l) =
;     { nil                          , l vida
;     { car(l) OR adev(cdr(l))          , altfel
;
; Parametri:
; l: lista de valori logice (T/NIL)
(defun adev(l)
    (cond
        ((null l) nil)
        (t (or (car l) (adev (cdr l))))
    )
)



; Model matematic:
; nivelL(l, e, niv) =
;     { true                         , l atom si l = e si niv % 2 = 0
;     { false                        , l atom si l = e si niv % 2 = 1
;     { false                        , l atom
;     { OR(nivelL(l1,e,niv+1),...,nivelL(ln,e,niv+1)) , altfel
;
; Parametri:
; l: arbore n-ar
; e: elementul cautat
; niv: nivelul curent
(defun nivelL (l e niv)
    (cond
        ((and (atom l)(equal l e)(equal (mod niv 2) 0)) t)
        ((and (atom l)(equal l e)(equal (mod niv 2) 1)) nil)
        ((atom l) nil)
        (t (funcall #'adev(mapcar #'(lambda (x) (nivelL x e (+ niv 1)))l)))
    )
)


; Model matematic:
; main2(l, e) = nivelL(l, e, -1)
;
; Parametri:
; l: arbore n-ar
; e: elementul cautat pe nivel par
(defun main2(L e)
    (nivelL L e -1)
)
(print (main2 '(a (b (g)) (c (d (e)) (f))) 'f))
```

```
;Un arbore n-ar se reprezintă în LISP astfel ( nod subarbore1 subarbore2 .....). Se cere să se determine calea de la radăcină
;către un nod dat. Se va folosi o funcție MAP.
;Exemplu pentru arborele (a (b (g)) (c (d (e)) (f)))
;a) nod=e => (a c d e)
;b) nod=v => ()


; Model matematic:
; cauta(l, e) =
;     { nil                          , l vida
;     { true                          , car(l) atom si car(l) = e
;     { cauta(car(l),e) OR cauta(cdr(l),e) , car(l) lista
;     { cauta(cdr(l),e)                   , altfel
;
; Parametri:
; l: arbore n-ar
; e: elementul cautat

(defun cauta (l e)
    (cond
        ((null l) nil)
        ((and (atom (car l))(equal (car l) e)) T)
        ((listp (car l)) (or (cauta (car l) e) (cauta (cdr l) e)))
        (t (cauta (cdr l) e))
    )
)


; drum(l, e) =
;     { e                            , l atom si l = e
;     { l                           , l atom
;     { append(car(l), drum(rest(l),e))  , l lista si cauta(l,e) = true
;     { nil                          , altfel
;
; Parametri:
; l: arbore n-ar
; e: nodul destinatie
(defun drum (l e)
    (cond
        ((and (atom l)(equal l e)) e)
        ((atom l) l)
        ((and (listp l) (equal (cauta l e) T) (funcall #'append (list (car l)) (mapcan #'(lambda (x) (drum x
e))l))))
        (t nil)
    )
)

(print (drum '(a (b (g)) (c (d (e)) (f))) 'e))



;D. Se consideră o listă neliniară. Să se scrie o funcţie care să aibă ca rezultat lista iniţială in care atomii de pe nivelul  k au fost
;înlocuiți cu 0 (nivelul superficial se consideră 1). Se va folosi o funcție MAP.
;Exemplu pentru lista (a (1 (2 b)) (c (d)))
```

;a) k=2 => (a (0 (2 b)) (0 (d))) b) k=1 => (0 (1 (2 b)) (c (d))) c) k=4 =>lista nu se modifică

```
; Model matematic:
; nivelK(l, k, niv) =
;     { 0                              , l atom si niv = k
;     { l                             , l atom
;     { nivelK(l1,k,niv+1) U ... U nivelK(ln,k,niv+1) , altfel
;
; Parametri:
; l: lista neliniara
; k: nivelul pe care se face inlocuirea
; niv: nivelul curent
(defun nivelK (l k niv)
   (cond
      ((and (atom l)(equal niv k)) 0)
      ((atom l) l)
      (t (mapcar #'(lambda (x) (nivelK x k (+ 1 niv)))l))
   )
)
(defun mainK (l k)
   (nivelK l k 0)
)
(print (mainK '(a (1 (2 b)) (c (d))) 2))
```

;D. Să se substituie valorile numerice cu o valoare e dată, la orice nivel al unei liste neliniare. Se va folosi o funcție MAP.
;Exemplu, pentru lista (1 d (2 f (3))), e=0 rezultă lista (0 d (0 f (0))).

```
; Model matematic:
; nivelNum(l, e) =
;     { e                              , l atom numeric
;     { l                             , l atom nenumeric
;     { nivelNum(l1,e) U ... U nivelNum(ln,e) , altfel
;
; Parametri:
; l: lista neliniara
; e: valoarea cu care se inlocuiesc numerele
(defun nivelNum (l e)
   (cond
      ((and (atom l) (numberp l)) e)
      ((atom l) l)
      (t (mapcar #'(lambda (x) (nivelNum x e))l))
   )
)
(defun mainNum (l e)
   (nivelNum l e)
)
(print (mainNum '(1 d (2 f (3))) 0))
```

;D. Se consideră o listă neliniară. Să se scrie o funcţie LISP care să aibă ca rezultat lista iniţială din care au

```
fost eliminaţi toţi
;atomii nenumerici de pe nivelurile pare (nivelul superficial se consideră 1). Se va folosi o funcție MAP.
;Exemplu pentru lista (a (1 (2 b)) (c (d))) rezultă (a (1 (2 b)) ((d)))


; Model matematic:
; nivelElim(l, niv) =
;     { [l]                          , l atom si niv % 2 = 1
;     { nil                          , l atom nenumeric si niv % 2 = 0
;     { [l]                          , l atom numeric
;     { nivelElim(l1,niv+1) U ... U nivelElim(ln,niv+1) , altfel
;
; Parametri:
; l: lista neliniara
; niv: nivelul curent

(defun nivelElim (l niv)
   (cond
      ((and (atom l) (equal (mod niv 2) 1)) (list l))
      ((and (atom l) (not (numberp l)) (equal (mod niv 2) 0)) nil)
      ((and (atom l) (numberp l)) (list l))
      (t (list (mapcan #'(lambda (x) (nivelElim x (+ niv 1)))l))) ; la feicare tura de mapcan pui list de ea
   )
)
(defun mainElim (l)
   (nivelElim l 0)
)
(print (mainElim '(a (1 (2 b)) (c (d)))))




;D. Un arbore n-ar se reprezintă în LISP astfel ( nod subarbore1 subarbore2 .....). Se cere să se determine
lista nodurilor de pe
;nivelurile pare din arbore (în ordinea nivelurilor 0, 2, …). Nivelul rădăcinii se consideră 0. Se va folosi o
funcție MAP.
;Exemplu pentru arborele (a (b (g)) (c (d (e)) (f))) => (a g d f)


; Model matematic:
; nivK(l, niv) =
;     { [l]                          , l atom si niv % 2 = 0
;     { nil                          , l atom
;     { nivK(l1,niv+1) U ... U nivK(ln,niv+1) , altfel
;
; Parametri:
; l: arbore n-ar
; niv: nivelul curent
(defun nivK (l niv)
   (cond
      ((and (atom l) (equal (mod niv 2) 0))(list l))
      ((atom l) nil)
      (t (mapcan #'(lambda (x) (nivK x (+ niv 1)))l))
   )
)
(defun mainLi (l)
   (nivK l -1)
```

```
)
(print (mainLi '(a (b (g)) (c (d (e)) (f)))))


;Un arbore n-ar se reprezintă în LISP astfel ( nod subarbore1 subarbore2 .....)
;Se cere să se înlocuiască nodurile de pe nivelul k din arbore cu o valoare e dată. Nivelul rădăcinii se consideră a fi 0.
;Se va folosi o funcție MAP.
;Exemplu pentru arborele (a (b (g)) (c (d (e)) (f))) şi e=h
;a) k=2 => (a (b (h)) (c (h (e)) (h)))
;b) k=4 => (a (b (g)) (c (d (e)) (f))


; Model matematic:
; nivel40(l, e, k, niv) =
;     { e                              , l atom si niv = k
;     { l                             , l atom
;     { nivel40(l1,e,k,niv+1) U ... U nivel40(ln,e,k,niv+1) , altfel
;
; Parametri:
; l: arbore n-ar
; e: valoarea de inlocuire
; k: nivelul tinta
; niv: nivelul curent
(defun nivel40(l e k niv)
    (cond
        ((and (atom l) (equal niv k)) e)
        ((atom l) l)
        (t (mapcar #'(lambda (x) (
            nivel40 x e k (+ niv 1)
        ))l))
    )
)
(defun main40 (l e k)
    (nivel40 l e k -1)
)
(print (main40 '(a (b (g)) (c (d (e)) (f))) 'h 2))


;Un arbore n-ar se reprezintă în LISP astfel ( nod subarbore1 subarbore2 .....). Se cere să se determine lista nodurilor de pe
;nivelurile pare din arbore (în ordinea nivelurilor 0, 2, …). Nivelul rădăcinii se consideră 0. Se va folosi o funcție MAP.
;Exemplu pentru arborele (a (b (g)) (c (d (e)) (f))) => (a g d f)


; elim(l niv) = { [l] , l e atom si niv % 2 = 0
; { [] , l e atom si niv % 2 = 1
; { elim(l2, niv+1) U ... U main(l2,niv+1) , altfel
; elim(l:list, niv:intreg)
(defun elim (l niv)
    (cond
        ((AND (atom l) (equal (mod niv 2) 0)) (list l))
        ((AND (atom l) (equal (mod niv 2) 1)) NIL)
        (T (mapcan #'(lambda (x) (elim x (+ niv 1))) l))
    )
)
```

```
; main(l) = elim(l,-1)
; main(l:list)
(defun main(l)
   (elim l -1)
)
(print (main '(a (b (g)) (c (d (e)) (f)))))


;Se consideră o listă neliniară. Să se scrie o funcţie care să aibă ca rezultat lista iniţială in care atomii de pe nivelurile pare
;au fost înlocuiți cu 0 (nivelul superficial se consideră 1). Se va folosi o funcție MAP.
;Exemplu pentru lista (a (1 (2 b)) (c (d))) se obține (a (0 (2 b)) (0 (d)))



; inlocuire(l, niv) = { 0 , l e atom si niv % 2 = 0
; { l , l e atom si niv % 2 = 1
; { inlocuire(l1,niv+1) U ... U inlocuire(ln,niv+1) , altfel (l e lista)
; inlocuire(l:list, niv:intreg)
(defun inlocuire(l niv)
   (cond
      ((AND (atom l) (equal 0 (mod niv 2))) 0)
      ((AND (atom l) (equal 1 (mod niv 2))) l)
      (T (mapcar #'(lambda (x) (inlocuire x (+ niv 1))) l))
   )
)
; main(l) = inlocuire(l, 0)
; main(l:list)
(defun main(l)
   (inlocuire l 0)
)

(print (main '(a (1 (2 b)) (c (d)))))



;D. Se dă o listă neliniară și se cere înlocuirea valorilor numerice impare situate pe un nivel par, cu numărul natural succesor.
;Nivelul superficial se consideră 1. Se va folosi o funcție MAP.
;Exemplu pentru lista (1 s 4 (3 f (7))) va rezulta (1 s 4 (4 f (7))).


; inlocuire(l, niv) = { l + 1 , l e atom, l e numar, l % 2 = 1, niv % 2 = 0
; { l , l e atom, l e numar, niv % 2 = 1
; { l , l e atom, l nu e numar
; { inlocuire(l1,niv+1) U ... U inlocuire(ln,niv+1) , altfel
; inlocuire(l:list, niv:intreg)
(defun inlocuire(l niv)
   (cond
      ((AND (atom l) (numberp l) (equal 1 (mod l 2)) (equal 0 (mod niv 2))) (+ l 1))
      ((AND (atom l) (numberp l) (equal 1 (mod niv 2))) l)
      ((AND (atom l) (not (numberp l))) l)
      (T (mapcar #'(lambda(x) (inlocuire x (+ niv 1))) l))
   )
)
; main(l) = inlocuire(l, 0)
; main(l:list)
```

```
(defun main(l)
    (inlocuire l 0)
)
(print (main '(1 s 4 (3 f (7)))))
```

;D. Se consideră o listă neliniară. Să se scrie o funcţie LISP care să aibă ca rezultat lista iniţială din care au fost eliminați toți
;atomii numerici multipli de 3. Se va folosi o funcție MAP.
;Exemplu a) dacă lista este (1 (2 A (3 A)) (6)) => (1 (2 A (A)) NIL)
;b) dacă lista este (1 (2 (C))) => (1 (2 (C)))

```
; elimina(l) = { NIL , l e atom numeric si l % 3 = 0
; { l , l e atom si l % 3 != 1 si l % 3 != 0
; { elimina(l1) U ... U elimina(ln) , altfel
; elimina(l:list)
(defun elimina(l)
    (cond
        ((AND (atom l) (numberp l) (equal 0 (mod l 3))) NIL)
        ((atom l) (list l))
        (T (list(mapcan #'elimina l)))
    )
)
; main(l) = elimina(l)[1]
; main(l:list)
(defun main(l)
    (car (elimina l))
)
(print (main '(1 (2 A (3 A)) (6))))
```

;D. Se dă o listă neliniară și se cere înlocuirea valorilor numerice care sunt mai mari decât o valoare k dată și sunt situate pe
;un nivel impar, cu numărul natural predecesor. Nivelul superficial se consideră 1. Se va folosi o funcție MAP.
;Exemplu pentru lista (1 s 4 (3 f (7))) și
;a) k=0 va rezulta (0 s 3 (3 f (6))) b) k=8 va rezulta (1 s 4 (3 f (7)))

```
; subs(l, k, niv) = { l - 1 , l e atom numeric si l > k si niv % 2 = 1
; { l , l e atom
; { subs(l1,k,niv+1) U ... U subs(ln,k,niv+1) , altfel
; subs(l:list, k:integer, niv:integer)
(defun subs(l k niv)
    (cond
        ((AND (atom l) (numberp l) (> l k) (equal 1 (mod niv 2))) (- l 1))
        ((atom l) l)
        (T (mapcar #'(lambda (x) (subs x k (+ niv 1))) l))
    )
)

; main(l, k) = subs(l, k, 0)
; main(l:list, k:integer)
(defun main(l k)
    (subs l k 0)
```

```
)

(print (main '(1 s 4 (3 f (7))) 0))   ; Test for k=0
(print (main '(1 s 4 (3 f (7))) 8))   ; Test for k=8



;D. Se consideră o listă neliniară. Să se scrie o funcţie LISP care să aibă ca rezultat lista iniţială din care au fost eliminați toți
;atomii numerici pari situați pe un nivel impar. Nivelul superficial se consideră a fi 1. Se va folosi o funcție MAP.
;Exemplu a) dacă lista este (1 (2 A (4 A)) (6)) => (1 (2 A (A)) (6))
;b) dacă lista este (1 (2 (C))) => (1 (2 (C)))


; elimina(l, niv) = { NIL , l e atom numeric si l % 2 = 0 si niv % 2 = 1
; { [l] , l e atom
; { elimina(l1,niv+1) U ... U elimina(ln,niv+1) , altfel
; elimina(l:list, niv:intreg)
(defun elimina(l niv)
    (cond
        ((AND (atom l) (numberp l) (equal 0 (mod l 2)) (equal 1 (mod niv 2))) NIL)
        ((atom l) (list l))
        (T (list (mapcan #'(lambda (x) (elimina x (+ niv 1))) l)))
    )
)

; main(l) = elimina(l,0)[1]
; main(l:list)
(defun main(l)
    (car (elimina l 0))
)
(print (main '(1 (2 A (4 A)) (6))))



;D. Se consideră o listă neliniară. Să se scrie o funcţie LISP care să aibă ca rezultat lista iniţială din care au fost eliminaţi toţi
;atomii de pe nivelul k (nivelul superficial se consideră 1). Se va folosi o funcție MAP.
;Exemplu pentru lista (a (1 (2 b)) (c (d)))
;a) k=2 => (a ((2 b)) ((d))) b) k=1 => ((1 (2 b)) (c (d))) c) k=4 =>lista nu se modifică


; elimina(l, niv, k) = { l , l atom si niv != k
; { [] , l atom si niv = k
; { elimina(l1, niv+1, k) U elimina(l2,niv,k+1) U ... U elimina(ln,niv,k+1) , altfel
; elimina(l:list, niv:intreg, k:intreg)
(defun elimina(l niv k)
    (cond
        (
            (AND (atom l) (not(equal niv k)))(list l))
            ((AND (atom l) (equal niv k))NIL)
            (T  (list (mapcan #'(lambda (x) (elimina x (+ niv 1) k))l)))))


; main(l, k) = elimina(l, 0, k)
```

```
; l - list
; k - intreg
(defun main(l k)
(car (elimina l 0 k))
)
(print (main '(a (1 (2 b)) (c (d))) 2))
(print (main '(a (1 (2 b)) (c (d))) 1))
(print (main '(a (1 (2 b)) (c (d))) 4))


;D. Un arbore n-ar se reprezintă în LISP astfel ( nod subarbore1 subarbore2 .....)
;Se cere să se înlocuiască nodurile de pe nivelurile impare din arbore cu o valoare  e dată. Nivelul rădăcinii se consideră a fi
;0. Se va folosi o funcție MAP.
;Exemplu pentru arborele (a (b (g)) (c (d (e)) (f))) şi e=h => (a (h (g)) (h (d (h)) (h)))


; Model matematic:
; inlocuireX(l, niv, e) =
;     { [e]                          , l atom si niv % 2 = 1
;     { [l]                         , l atom
;     { inlocuireX(l1,niv+1,e) U ... U inlocuireX(ln,niv+1,e) , altfel
;
; Parametri:
; l: arbore n-ar
; niv: nivelul curent
; e: valoarea de inlocuire
(defun inlocuireX (l niv e)
   (cond
      ((and (atom l) (equal (mod niv 2) 1)) (list e))
      ((atom l) (list l))
      (t (list (mapcan #'(lambda (x) (inlocuireX x (+ niv 1) e))l)))
   )
)
(print (inlocuireX '(a (b (g)) (c (d (e)) (f))) -1 'h))



;D. Se dă o listă neliniară și se cere înlocuirea valorilor numerice pare cu numărul natural succesor.  Se va folosi o funcție
;MAP.
;Exemplu pentru lista (1 s 4 (2 f (7))) va rezulta (1 s 5 (3 f (7))).


; Model matematic:
; succesor(l) =
;     { [l + 1]                      , l atom numeric si l % 2 = 0
;     { [l]                         , l atom
;     { succesor(l1) U ... U succesor(ln) , altfel
;
; Parametri:
; l: lista neliniara
(defun succesor (l)
   (cond
      ((and (atom l) (numberp l)(equal (mod l 2) 0)) (list (+ l 1)))
      ((atom l) (list l))
```

```
        (t (list (mapcan #'(lambda (x) (succesor x))l)))
    )
)
(print (succesor '(1 s 4 (2 f (7)))))


;D. Un arbore n-ar se reprezintă în LISP astfel (nod subarbore1 subarbore2 .....). Se cere să se determine numărul de noduri
;de pe nivelul k. Nivelul rădăcinii se consideră 0. Se va folosi o funcție MAP.
;Exemplu pentru arborele (a (b (g)) (c (d (e)) (f)))
;a) k=2 => nr=3 (g d f) b) k=4 => nr=0 ()


; Model matematic:
; numarNod(l, k, niv) =
;     { 1                          , l atom si niv = k
;     { 0                          , l atom
;     { suma(numarNod(l1,k,niv+1),...,numarNod(ln,k,niv+1)) , altfel
;
; Parametri:
; l: arbore n-ar
; k: nivelul dorit
; niv: nivelul curent
(defun numarNod (l k niv)
    (cond
        ((and (atom l)(equal niv k)) 1)
        ((atom l) 0)
        (t (apply #'+ (mapcar #'(lambda (x) (numarNod x k (+ niv 1)))l)))
        )
    )
(print (numarNod '(a (b (g)) (c (d (e)) (f))) 2 -1))


;D. Se consideră o listă neliniară. Să se scrie o funcţie LISP care să aibă ca rezultat lista iniţială din care au fost eliminate toate
;aparițiile unui element e. Se va folosi o funcție MAP.
;Exemplu a) dacă lista este (1 (2 A (3 A)) (A)) şi e este A => (1 (2 (3)) NIL)
;b) dacă lista este (1 (2 (3))) şi e este A => (1 (2 (3)))



; Model matematic:
; eliminaE(l, e) =
;     { nil                        , l atom si l = e
;     { [l]                        , l atom
;     { eliminaE(l1,e) U ... U eliminaE(ln,e) , altfel
;
; Parametri:
; l: lista neliniara
; e: elementul de eliminat
(defun eliminaE(l e)
    (cond
        ((and (atom l)(equal l e)) nil)
        ((atom l) (List l))
        (t (list(mapcan #'(lambda (x) (eliminaE x e))l)))
    )
)
```

```
(print (eliminaE '(1 (2 A (3 A)) (A)) 'A))


;D. Un arbore n-ar se reprezintă în LISP astfel (nod subarbore1 subarbore2 .....). Se cere să se determine lista nodurilor de pe
;nivelul k. Nivelul rădăcinii se consideră 0. Se va folosi o funcție MAP.
;Exemplu pentru arborele (a (b (g)) (c (d (e)) (f)))
;a) k=2 => (g d) b) k=5 => ()


; Model matematic:
; depeK(l, k, niv) =
;     { [l]                          , l atom si niv = k
;     { nil                          , l atom
;     { depeK(l1,k,niv+1) U ... U depeK(ln,k,niv+1) , altfel
;
; Parametri:
; l: arbore n-ar
; k: nivelul dorit
; niv: nivelul curent
(defun depeK (l k niv)
   (cond
      ((and (atom l)(equal k niv)) (list l))
      ((atom l) nil)
      (t (mapcan #'(lambda (x) (depeK x k (+ niv 1)))l))
   )
)
(print (depeK '(a (b (g)) (c (d (e)) (f))) 2 -1))
```