2. rezolvare în .jpg

3. Evitarea apelului recursiv - LISP

```
(defun F (e)
    (cond
        ((null e) nil)
        ((listp (car e)) ((lambda (x)
                            (append x (F (cdr e)) (car x))
                          )
                          (F (car e))
                         )
        )
        (t (list (car e)))
    )
)
```

4. Eliminarea tuturor aparițiilor elementului x.

$$elimina(e,x) = \begin{cases} (e), & e \text{ e atom și } e \neq x \\ \phi, & e = x \\ \bigcup_{i=1}^{m} elimina(e_i), & altfel \end{cases}$$

$main(e,x) = y_1$, unde $elimina(e,x) = y$ și $y = y_1 \dots y_m$

```
(defun elimina (e x)
    (cond
        ((and (atom e) (equal e x)) nil)
        ((atom e) (list e))
        (t (list (mapcan #'(lambda (y)
                             (elimina y x)
                           )
                 e
                 )
           )
        )
    )
)
```

; s-a utilizat o functie ajutatoare, deoarece
; nu s-ar fi putut apela corect functia 4 eliminari
; aceasta avandu-e ca argument si pe x, var. care
; ramane constanta pe tot parcursul algoritmului

)
)

(defun main(l x)
(car (eliminca l x)))

5. Nr. de sublisti pt. care suma atomilor numerici de pe nivelurile impare este nr. par.
(nivelul superficial se considera 1)

Modelul matematic:

$$suma(e, niv) = \begin{cases} 0, & e \text{ e atom nenumeric} \\ 0, & e \text{ e nr. si niv e par} \\ e, & e \text{ e nr. si niv e impar} \\ \sum_{i=1}^{n} suma(e_i, niv+1), & altfel \end{cases}$$

$$sublisti(e) = \begin{cases} 0, & e \text{ e atom} \\ 1 + \sum_{i=1}^{n} sublisti(e_i), & dacă\ suma(e,0) \text{ e nr. par} \\ \sum_{i=1}^{n} sublisti(e_i), & altfel \end{cases}$$

reprezintă "nivelul" din care face lista => niv. superficial va fi 1

(defun suma(l niv)
(cond
((and (numberp l) (oddp niv)) l)
((and (numberp l) (evenp niv)) nil)

```
((atom e) nil)
(+ (apply #'+ (mapcar #'(lambda (y)
                          (suma y (+ nil 1)))
                        e)))
)
)
)

(defun sucesiste (e)
  (cond
    ((atom e) 0)
    ((evenp (suma e 0)) (+ 1 (apply #'+ (
      mapcar #'sucesiste e))))
    (+ (apply #'(mapcar #'sucesiste e)))
  )
)
```