1. La prima vedere, funcția ar fi trebuit să returneze produsul numerelor pare din listă, însă există 2 probleme:

1. produsul este inițializat cu 0
2. (cea mai mare problemă) cut-ul din a doua ramură ne permite elementelor să ajungă și pe a 3-a ramură
   => dacă elementul curent este impar, acesta fiind automat alocat în a doua ramură, nu va putea îndeplini condiția de nr. par

=> Pentru a evita ca nr. impare, predicatul va da întotdeauna fals roșu.

Rezultatul apelului f([1,2,3,4,5],S) este fals roșu.

Modificare:

```
f([],0).
f([H|T],S) :- H mod 2 =:= 0, !, f(T,S1),
              S is S1+1.
f([_|T],S) :- f(T,S).
```

4. Inlănțuirea atomilor de pe nivel impar cu un element dat. (niv. rădăcinii = 0).

Modele matematice:

$$\text{inlantuire}(e, x, niv) = \begin{cases} x, & e \text{ e atom și niv impar} \\ e, & e \text{ e atom și niv par} \\ \bigcup_{i=1}^{n} \text{inlantuire}(e_i, x, niv+1), & \text{altfel} \end{cases}$$

$$main(e, x) = inlantuire(e, x, -1)$$

```
(defun incacuire (e x niv)
    (cond
        ((and (atom e) (oddp niv)) x)
        ((atom e) e)
        (t (mapcar #'(lambda (y)
                       (incacuire y x (+ niv 1)))
                   e)
        )
    )
)

(defun main (e x)
    (incacuire e x -1)
)
```

5. Nr. de sublisteste pt. care atomul numeric maxim este par.

Modele matematice:

$$maxim(e_1 \dots e_n, m) = \begin{cases} m, & n = 0 \\ maxim(e_2 \dots e_n, e_1), & e_1 \text{ e nr. și } e_1 > m \\ maxim(e_2 \dots e_n, m), & e_1 \text{ e nr. și } e_1 \leq m \text{ SAU } e_1 \text{ e atom nenumeric} \\ maxim(e_2 \dots e_n, y), & \text{altfel} \end{cases}$$

unde $y = maxim(e_1, m)$

$$nr(e) = \begin{cases} 0, & e \text{ e atom} \\ 1 + \sum_{i=1}^{n} nr(e_i), & pf.\ maxim(e, -\infty) \text{ e nr. par} \\ \sum_{i=1}^{n} nr(e_i), & \text{altfel} \end{cases}$$

```
(defun maxim (e m)
    (cond
        ((null e) m)
        ((and (numberp (car e)) (> (car e) m)
            (maxim (cdr e) (car e)))
        ((atom (car e)) (maxim (cdr e) m))
        (t (maxim (cdr e) (maxim (car e) m)))
    )
)

(defun nr (e)
    (cond
        ((atom e) 0)
        (((lambda (x)
            (and (not (equal x most-negative-fixnum))
                 (evenp x)))
          (maxim e most-negative-fixnum))
         (+ 1 (apply #'+ (mapcar #'nr e))))
        (t (apply #'+ (mapcar #'nr e)))
    )
)
```