

Programare logică și funcțională

- examen scris -

Notă

1. Subiectele se notează astfel: of - 1p; A – 2p; B - 4p; C - 3p.
2. Problema Prolog (B) vor fi rezolvată în SWI Prolog. Se cere: (1) explicarea codului și a raționamentului; (2) modelul recursiv de rezolvare, pentru fiecare predicat folosit; (3) specificarea fiecărui predicat (semnificația parametrilor, model de flux, tipul predicatului - determinist/nedeterminist).
3. Problema Lisp (C) va fi rezolvată în Common Lisp. Se cere: (1) explicarea codului și a raționamentului; (2) modelul recursiv de rezolvare, pentru fiecare funcție folosită; (3) specificarea fiecărei funcții (semnificația parametrilor).

A. Fie **G** o funcție LISP și fie următoarea definiție

```
(DEFUN F(L)
  (COND
    ((NULL L) 0)
    (> (G L) 2) (+ (G L) (F (CDR L))))
    (T (G L))
  )
)
```

Rescrieți această definiție pentru a evita apelul repetat **(G L)**, fără a redefini logica clauzelor și fără a folosi o funcție auxiliară. Nu folosiți SET, SETQ, SETF. Justificați răspunsul.

B. Dându-se o listă formată din numere întregi, să se genereze în PROLOG lista aranjamentelor cu număr par de elemente, având suma număr impar. Se vor scrie modelele matematice și modelele de flux pentru predicatele folosite.

Exemplu- pentru lista $L=[2,3,4] \Rightarrow [[2,3],[3,2],[3,4],[4,3]]$ (nu neapărat în această ordine)

C. Un arbore n-ar se reprezintă în LISP astfel (nod subarbore1 subarbore2). Se cere să se determine înălțimea unui nod în arbore. **Se va folosi o funcție MAP.**

Exemplu pentru arborele (a (b (g)) (c (d (e)) (f)))

a) nod=e => înălțimea e 0 **b)** nod=v => înălțimea e -1 **c)** nod=c => înălțimea e 2