

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 următoarea definiție de funcție LISP

```
(DEFUN F(L)
  (COND
    ((NULL L) 0)
    ((> (F (CAR L)) 1) (F (CDR L)))
    (T (+ (F (CAR L)) (F (CDR L)))))
)
```

Rescrieți această definiție pentru a evita dublul apel recursiv (**F (CAR L)**), fără a redefini logica clauzelor și fără a folosi o funcție auxiliară. Nu folosiți SET, SETQ, SETF. Justificați răspunsul.

B. Să se scrie un program PROLOG care generează lista aranjamentelor de **k** elemente dintr-o listă de numere întregi, având o sumă **S** dată. Se vor scrie modelele matematice și modelele de flux pentru predicatele folosite.

Exemplu- pentru lista [6, 5, 3, 4], **k**=2 și **S**=9 \Rightarrow [[6,3],[3,6],[5,4],[4,5]] (nu neapărat în această ordine)

C. 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))) și **e=h** => (a (h (g)) (h (d (h)) (h)))