

Programare logică și funcțională

- examen scris -

Notă

1. Subiectele se notează astfel: of - 1p; A - 1.5p; B - 2.5p; C - 2.5p; D - 2.5p.
2. Problemele Prolog vor fi rezolvate 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. Problemele Lisp vor fi rezolvate 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 în LISP

```
(DEFUN F(L1 L2)
  (APPEND (F (CAR L1) L2)
    (COND
      ((NULL L1) (CDR L2))
      (T (LIST (F (CAR L1) L2) (CAR L2)))
    )
  )
)
```

Rescrieți această definiție pentru a evita dublul apel recursiv **(F (CAR L1) L2)**. Nu redefiniți funcția. Nu folosiți SET, SETQ, SETF. Justificați răspunsul.

- B.** Dându-se un arbore binar în care nodurile conțin informații numerice și arborele este reprezentat sub forma unei liste în care fiecare nod este urmat de un număr (0, 1 or 2) care reprezintă numărul de descendenți ai nodului respectiv, se cere un program SWI-Prolog care calculează suma primului element de pe fiecare nivel. De exemplu, pentru lista [13, 2, 9, 2, 5, 0, 3, 2, 11, 0, 6, 1, 3, 0, 2, 1, 7, 1, 9, 1, 8, 2, 4, 0, 2, 1, 10, 0] rezultatul va fi 55.

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

Exemplu- pentru lista [2, 5, 3, 4, 10], **k**=2 și **P**=20 \Rightarrow [[2,10],[10,2],[5,4],[4,5]] (nu neapărat în această ordine)

- D. Un arbore n-ar se reprezintă în LISP astfel (nod subarbore1 subarbore2). Se cere să se determine calea de la rădăcină 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 => ()