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 definiţie de funcţie LISP

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

Rescrieți această definiție pentru a evita apelul recursiv repetat **(F (CDR L))**. Nu redefiniți funcția. Nu folosiți SET, SETQ, SETF. Justificați răspunsul.

B. Dându-se o listă eterogenă formată din numere și liste liniare de numere, se cere un program SWI-PROLOG care să calculeze diferența dintre cel mai mare număr din subliste și cel mai mic număr de la nivelul superficial al listei. Se va presupune că lista de intrare conține cel puțin o sublistă și cel puțin un număr la nivel superficial, dar nu se cunoaște valoarea minimă/maximă posibilă pentru numerele din listă/subliste. **De exemplu**, pentru lista [[4, 2, 8], 7, 2, -3, [6, 9, 11, 2], 4], rezultatul va fi 14 [11 - [-3]].

C. Scrieţi un program PROLOG care determină dintr-o listă formată din numere întregi lista subşirurilor cu cel puțin 2 elemente, formate din elemente în ordine strict crescătoare. Se vor scrie modelele matematice și modelele de flux pentru predicatele folosite.

Exemplu- pentru lista [1, 8, 6, 4] $\Rightarrow$ [[1,8],[1,6],[1,4],[6,8],[4,8],[4,6],[1,4,6],[1,4,8],[1,6,8],[4,6,8],[1,4,6,8]] (nu neapărat în această ordine)

D. Se consideră o listă neliniară. Să se scrie o funcţie LISP care să aibă ca rezultat lista iniţială în care toate apariţiile unui element **e** au fost înlocuite cu o valoare **e1**. **Se va folosi o funcție MAP.**

Exemplu **a)** dacă lista este (1 (2 A (3 A)) (A)) **e** este A şi **e1** este B => (1 (2 B (3 B)) (B))

b) dacă lista este (1 (2 (3))) şi **e** este A => (1 (2 (3)))