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 predicat PROLOG **f(integer, integer),** având modelul de flux (i, o):

```
f(50, 1):-!.
f(I,Y):-J is I+1, f(J,S), S<1, !, K is I-2, Y is K.
f(I,Y):-J is I+1, f(J,Y).
```

Rescrieți această definiție pentru a evita apelul recursiv **f(J,V)** în ambele clauze. Nu redefiniți predicatul. Justificați răspunsul.

B. Dându-se o listă neliniară conținând atât atomi numerici, cât și nenumerici, se cere un program LISP care să înlocuiască fiecare atom nenumeric cu numărul de apariții ale atomului la nivelul pe care se află. **De exemplu**, pentru lista (F A 12 13 (B 11 (A D 15) C C (F)) 18 11 D (A F) F), rezultatul va fi (2 1 12 13 (1 11 (1 1 15) 2 2 (1)) 18 11 1 (1 1) 2).

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 şi **P**=20 ⇒ [[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 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))) şi **e**=h => (a (h (g)) (h (d (h)) (h)))