Functional and logic programming
- **written exam -**

Important:

1. Subjects are graded as follows: of - 1p; A – 1.5p; B - 2.5p; C - 2.5p; D - 2.5p.
2. Prolog problems will be resolved using SWI Prolog. The following are required: (1) explanation of the code and of the reasoning behind it; (2) recursive model that solves the problem, for all the predicates used; (3) specification of every predicate (parameters and their meaning, flow model, type of the predicate - deterministic/non-deterministic).
3. Lisp problems will be resolved using Common Lisp. The following are required: (1) explanation of the code and of the reasoning behind it; (2) recursive model that solves the problem, for each function used; (3) specification of every function (parameters and their meaning).

A. Given the following PROLOG predicate definition **f(integer, integer),** with the flow model (i, o):

```
f(100, 1):-!.
f(K,X):-K1 is K+1, f(K1,Y), Y>1, !, K2 is K1-1, X is K2+Y.
f(K,X):-K1 is K+1, f(K1,Y), Y>0.5, !, X is Y.
f(K,X):-K1 is K+1, f(K1,Y), X is Y-K1.
```

Rewrite the definition in order to avoid the recursive call **f(J,V)** in all clauses. Do NOT redefine the predicate. Justify your answer.

B. Given a nonlinear list of both numerical and non-numerical atoms, write a LISP program that builds a linear list composed only from those non-numerical atoms that occur an even number of times in the initial list. The result will contain each element only once, in reverse order of the initial list. **For example**, for the list (F A 2 3 (B 1 (A D 5) C C (F)) 8 11 D (A F) F), the result will be (C D F). You are NOT allowed to use predefined LISP functions *reverse* or *member*.

C. Given a list composed of integer numbers, generate in PROLOG the list of arrangements of N elements ending with an odd value and have the sum S given. Write the mathematical models and flow models for the predicates used. For example, for the list L=[2,7,4,5,3], N=2 and S=7 ⇒ [[2,5], [4,3]] (not necessarily in this order).

D. An n-ary tree is represented in Lisp as (node subtree1 subtree2 ...). Write a Lisp function to verify whether a node **x** occurs on an even level of the tree. The root level is assumed zero. **A MAP function shall be used.**

Example for the tree (a (b (g)) (c (d (e)) (f))) **a)** **x**=g => T **b)** x=h => NIL