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. Let L be a list of numbers and given the following PROLOG predicate definition with flow model (i, o):

```
f([],-1).
f([H|T],S):-f(T,S1),S1>0,!,S is S1+H.
f([_|T],S):-f(T,S1),S is S1.
```

Rewrite the definition in order to avoid the recursive call **f(T,S)** in both clauses. Do NOT redefine the predicate. Justify your answer.

B. Given a nonlinear list containing numerical and non-numerical atoms, write a LISP program that verifies if the numerical atoms in the list form an increasing sequence. For example, for the list (A B 1 (2 C D) 3 4 (F T 6 10 (A E D) (34) F) 111)) the result will be **true** (T), and for the list (A B 1 (2 C D) 3 4 (F T 6 1 (A E D) (34) F) 111)) the result will be **false** (NIL).

C. Given a list made of integer numbers, generate in PROLOG the list of all subsets with even number of elements. Write the mathematical models and flow models for the predicates used. For example, for the list L=[2,3,4] $\Rightarrow$ [[],[2,3],[2,4],[3,4]] (not necessarily in this order).

D. An n-ary tree is represented in Lisp as (node subtree1 subtree2 ...). Write a Lisp function to determine the number of nodes on level **k**. The root level is assumed zero. **A MAP function shall be used.** ***Example*** for the tree (a (b (g)) (c (d (e)) (f)))
a) k=2 => nr=3 (g d f) **b)** k=4 => nr=0 ()