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(0, -1):-!.
f(I,Y):-J is I-1, f(J,V), V>0, !, K is J, Y is K+V.
f(I,Y):-J is I-1, f(J,V), Y is V+I.
```

Rewrite the definition in order to avoid the recursive call **f(J,V)** 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. Write a PROLOG program that generates the list of all combinations of k elements with the value of sum of each combination even number, from a list of integers. Write the mathematical models and flow models for the predicates used. For example, for the list L[6, 5, 3, 4], **k**=2 ⇒ [[6,4],[5,3]] (not necessarily in this order).

D. Given a nonlinear list, write a Lisp function to return the list with all atoms on level **k** removed. The superficial level is assumed 1. **A MAP function shall be used.**
Example for the list (a (1 (2 b)) (c (d)))
a) k=2 => (a ((2 b)) ((d))) **b)** k=1 => ((1 (2 b)) (c (d))) **c)** k=4 => the list does not change