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 both numerical and non-numerical atoms, write a LISP program that computes the greatest common divisor of odd numbers on even levels of the list. The superficial level is considered to be 1. For example, for the list (A B 12 (9 D (A F (75 B) D (45 F) 1) 15) C 9), the result will be 3. We assume there is at least one odd number at each even level of the list. You are not allowed to use the predefined Lisp function *gcd*.

C. Write a PROLOG program that generates the list of all subsets, each subset having an odd sum of elements and also even number of elements. Write the mathematical models and flow models for the predicates used. For example, for [2,3,4] ⇒ [[2,3,4]].

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