Functional and logic programming
- written exam -

<u>Important:</u>

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. The following function definition in LISP is given

```
(DEFUN F(L)
        (COND
              ((NULL L) 0)
              (> (F (CDR L)) 2) (+ (F (CDR L)) (CAR L)))
              (T (+ (F (CDR L)) 1))
        )
)
```

Rewrite the definition in order to avoid the repeated recursive call **<u>(F (CDR L))</u>**. Do NOT redefine the function. Do NOT use SET, SETQ, SETF. Justify your answer.

B. Given a list composed of integer numbers and sublists of integer numbers, write a SWI-Prolog program that verifies if all the elements of the list (including those in sublists) form a symmetrical sequence. For example, for the list [1, 5, [2,4], 7, 11, 25, [11, 7, 4], 2, 5, 1] the result will be true.

C. Write a PROLOG program that generates the list of all subsets with value of sum for each subset odd number and also odd numbers of odd values from each subset. Write the mathematical models and flow models for the predicates used. For example, for [2,3,4] ⇒ [[2,3],[3,4],[2,3,4]] not necessarily in this order).

D. Given a nonlinear list, write a Lisp function to return the list with all atoms on level **k** replaced by **0**. 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 (0 (2 b)) (0 (d)))
(b) k=1 => (0 (1 (2 b)) (c (d)))
(c) k=4 => the list does not change