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

**Important:**

1. Subjects are graded as follows: By default - 1p; A – 2p; B - 4p; C - 3p.
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(L1 L2)
        (APPEND (F (CAR L1) L2)
                (COND
                    ((NULL L1) (CDR L2))
                    (T (LIST (F (CAR L1) L2) (CAR L2)))
                )
                )
        )
```

Rewrite the definition in order to avoid the double recursive call **(F (CAR L1) L2)**. Do NOT redefine the function. Do NOT use SET, SETQ, SETF. Justify your answer.0

**B.** Write a PROLOG program that generates the list of all subsets of k elements in arithmetic progression. Write the mathematical models and flow models for the predicates used. For example, for L=[1,5,2,9,3] and k=3 ⇒ [[1,2,3],[1,5,9],[1,3,5]] (not necessarily in this order).

**C.** 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
**C.** Given a nonlinear list, write a Lisp function to return the list with all occurrences of an element **e** removed. **A MAP function shall be used.**
***Example*** **a)** if the list is (1 (2 A (3 A)) (A)) and **e** is A => (1 (2 (3)) NIL)
**b)** if the list is (1 (2 (3))) and **e** is A => (1 (2 (3)))