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. The following function definition in LISP is given

```
(DEFUN F(N)
        (COND
                ((= N 0) 0)
                (> (F (- N 1)) 1) (- N 2))
                (T (+ (F (- N 1)) 1))
        )
)
```

Rewrite the definition in order to avoid the double recursive call **(F (- N 1))**. Do NOT redefine the function. Do NOT use SET, SETQ, SETF. Justify your answer.

B. Given a heterogeneous list composed of numbers and nonempty linear numerical lists, write a SWI-Prolog program that inverts the sublists for which the lowest common multiple of the elements is greater than the square of the first element of the sublist. For example, for the list [[4, 1, 18], 7, 2, -3, [6, 9, 11, 3], 4, [9, 4, 3]], the correct result is: [[18, 1, 4], 7, 2, -3, [3, 11, 9, 6], 4, [9, 4, 3]].

C. For a given value N, generate the list of all permutations with elements N, N+1, ..., 2*N-1 with the property that the absolute value between two consecutive values from the permutation is <=2. Write the mathematical models and flow models for the predicates used.

D. Given a nonlinear list, write a Lisp function to return the list with all even numerical atoms from an odd level removed. The superficial level is assumed 1. **A MAP function shall be used.**
Example **a)** if the list is (1 (2 A (4 A)) (6)) => (1 (2 A (A)) (6))
b) if the list is (1 (2 (C))) => (1 (2 (C)))