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. Let L be a list of numbers and given the following PROLOG predicate definition with flow model (i, o):

```
f([],-1).
f([H|T],S):-f(T,S1), S1<1, S is S1-H, !.
f([_|T],S):-f(T,S).
```

Rewrite the definition in order to avoid the recursive call **f(T,S)** in both clauses. Do NOT redefine the predicate. Justify your answer.

B. Given a nonlinear list that contains numerical and non-numerical atoms, write a Lisp program that verifies if the sequence of the numerical atoms on all odd levels form a zig-zag sequence (element 2 is greater than the first element, element 3 is smaller than element 2, element 4 is greater than element 3, etc.). For example, for the list (10 21 (3 A (B (0 77) 1 77)) C (5 (D 54) 11 6) 89 F H) the result will be true (the zig-zag sequence is (10 21 1 77 54 89)).

C. Write a PROLOG program that generates the list or all arrangements of **k** elements from a list of integer numbers, for which the product of the elements is less than a value **V** given. Write the mathematical models and flow models for the predicates used. For example, for the list [1, 2, 3], **k**=2 and **V**=7 ⇒ [[1,2],[2,1],[1,3],[3,1],[2,3],[3,2]] (not necessarily in this order).

D. An n-ary tree is represented in Lisp as (node subtree1 subtree2 ...). Write a Lisp program to return the ***height*** of a node of a tree. **A MAP function shall be used.**
Example for the tree (a (b (g)) (c (d (e)) (f)))
a) nod=e => the height is 0 **b)** nod=v => the height is -1 c**)** nod=c => the height is 2.