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(G L)
        (COND
                ((NULL L) NIL)
                (> (FUNCALL G L) 0) (CONS (FUNCALL G L) (F (CDR L))))
                (T (FUNCALL G L))
        )
)
```

Rewrite the definition in order to avoid the repeated call **(FUNCALL G L)**. Do NOT redefine the function. Do NOT use SET, SETQ, SETF. Justify your answer.

B. Given two lists composed of integer numbers and sublists of integer numbers, write a SWI-Prolog program that returns a list that contains, for each possible pair of sublists (one sublist from the first list and one from the second), the product of the maximum elements. For example, for the following two lists of sublists [1,2, [4,2], 6, [3,2]] and [1,2,3,[5,6],8, 5,[12,3], 4,1,[3,8]] the result will be (not necessarily in this order): [24, 48, 32, 18, 36, 24].

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).

```
1  insertE(E, L, [E|L]).
2  insertE(E, [H|T], [H|L]):-
3      insertE(E, T, L).
4
5  arr([E|_], 1, [E]).
6  arr([_|T], K, L):-
7      arr(T, K, L).
8  arr([H|T], K, R1):-
9      K > 1,
10     K1 is K - 1,
11     arr(T, K1, R),
12     insertE(H, R, R1).
13
14 prod([], 1).
15 prod([H|T], P):-
16     prod(T, P1),
17     P is P1 * H.
18
19 main(L, K, P, Res):-
20     arr(L, K, Arr),
21     prod(Arr, Prod),
22     Prod =< P,
23     Res = Arr.
```

D. Given a nonlinear list, write a Lisp function to return the list with all non-numerical atoms on even levels removed. The superficial level is assumed 1. **A MAP function shall be used.** ***Example*** for the list (a (1 (2 b)) (c (d))) the result is (a (1 (2 b)) ((d)))

```
(defun removeK (L K)
  (cond
    ((atom L)
      (cond
        ((and (= 0 (mod K 2)) (not (numberp L))) nil)
        (t (list L))
      )
    )
    (t (list (apply #'append (mapcar #'(lambda (x) (removeK x (+ K 1))) L))))
  )
)


(defun remv(L)
  (car (removeK L 0))
)

(print (remv '(a (1 (2 b)) (c (d)))))
```