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 numeric linear list, write a SWI-PROLOG program that returns (as a list of pairs) all possible partitions of the initial list in two sublists, such that the average value of the elements from the first sublist is smaller or equal to the greatest common divisor of the elements from the second sublist. **For example**, for the list [8, 4, 6, 1], the result will be (not necessarily in this order): [[[1, 4, 8], [6]], [[1, 6, 4], [8]], [[1, 6], [4, 8]], [[1], [6, 4, 8]]].

```
44 partitios([],[],[]).
45 partitios([H|T], L1, [H|L2]):-
46     partitios(T, L1, L2).
47 partitios([H|T], [H|L1], L2):-
48     partitios(T, L1, L2).
49
50 list_gcd([H|T], GCD) :-
51     foldl(gcd, T, H, GCD).
52
53 gcd(X, 0, X) :- X > 0.
54 gcd(X, Y, G) :-
55     Y > 0,
56     R is X mod Y,
57     gcd(Y, R, G).
58
59 average(L, A):-
60     suma(L, S),
61     length(L, Len),
62     Len > 0,
63     A is S / Len.
64
65 oneSol(L, R):-
66     partitios(L, L1, L2),
67     L1 \= [],
68     L2 \= [],
69     average(L1, A),
70     list_gcd(L2, Gcd),
71     A =< Gcd,
72     R = [L1, L2].
73
74 allSol(L, R):-
75     findall(Res, (oneSol(L, Res)), R).
```

C. Write a PROLOG program that generates the list of all subsets of sum **S** given, using the elements of a list, such that the number of even elements from each subset is even. Write the mathematical models and flow models for the predicates used. For example for the list [1, 2, 3, 4, 5, 6, 10] and S=10 ⇒ [[1,2,3,4], [4,6]].

```
30 subS([], []).
31 subS([_|T], L):-
32     subS(T, L).
33 subS([H|T], [H|L]):-
34     subS(T, L).
35
36 suma([], 0).
37 suma([H|T], R):-
38     suma(T, R1),
39     R is R1 +H.
40
41 cntEv([],0).
42 cntEv([H|T], R):-
43     H mod 2 =:= 0,!,
44     cntEv(T, R1),
45     R is R1 + 1.
46 cntEv([_|T], R):-
47     cntEv(T, R).
48
49 oneSol(L, S, R):-
50     subS(L, SubS),
51     suma(SubS, Sum),
52     Sum =:= S,
53     cntEv(SubS, Cnt),
54     Cnt mod 2 =:= 0,
55     R = SubS.
56
57 allSol(L, S, R):-
58     findall(Res, oneSol(L, S, Res), R).
59 % allSol([1,2,3,4,5,6,10], 10, R)
```

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.

```
(defun hei(e tree found)
  (cond
    ((atom tree) -1)
    ((and (listp tree) (equal found nil) (not (equal e (car tree))))
      (apply #'max (mapcar (lambda(a) (hei e a nil)) tree)))
    ((and (listp tree) (equal found nil) (equal e (car tree)))
      (+ 1 (apply #'max (mapcar (lambda(a) (hei e a t)) tree))))
    (t (+ 1 (apply #'max (mapcar (lambda(a) (hei e a t)) tree))))
  )
)
(print (hei 'd '(a (b (g)) (c (d (e)) (f))) nil))
```