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).

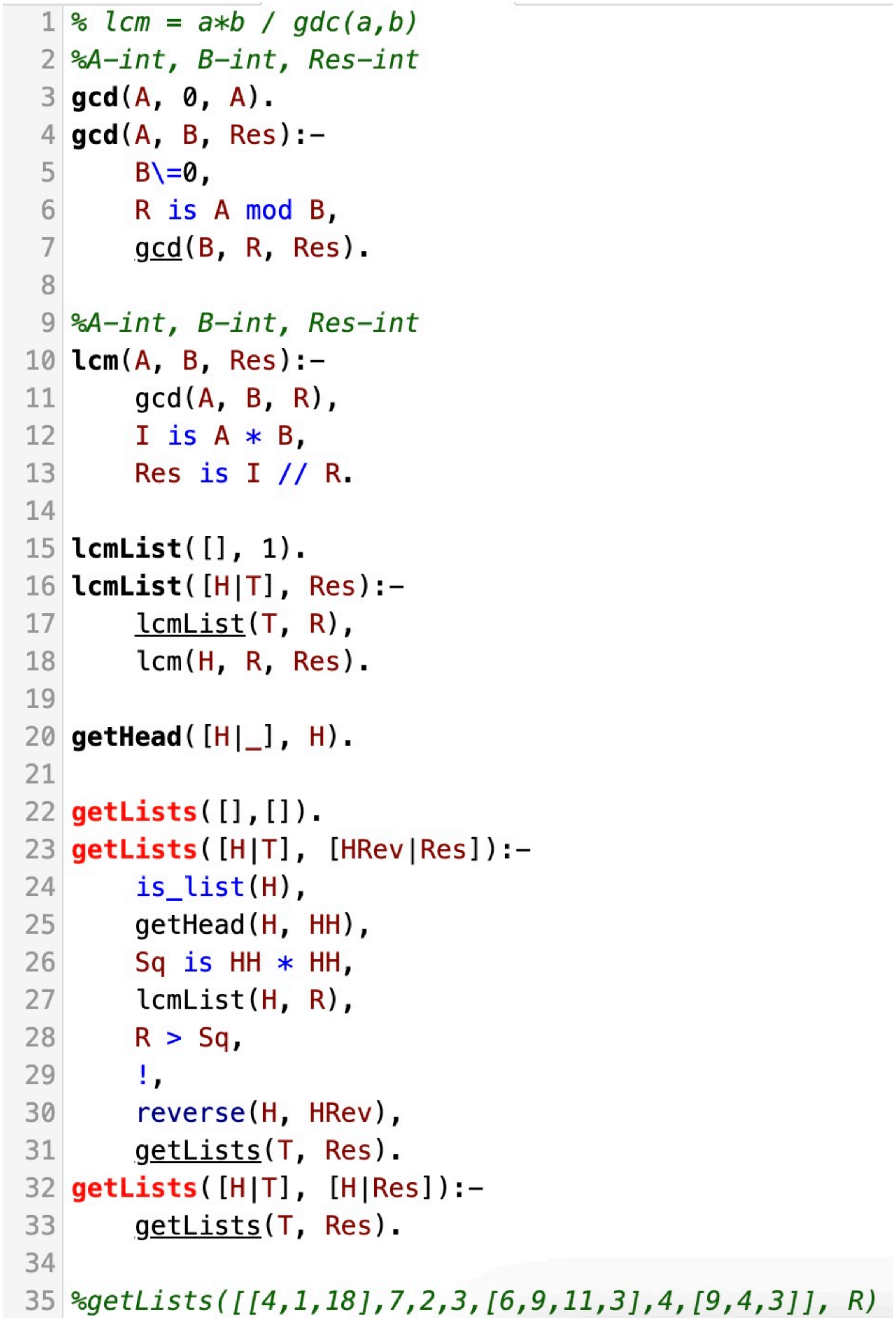
A. The following function definition in LISP is given

```
(DEFUN F(L)
    (COND
        ((NULL L) NIL)
        (> (F (CAR L)) 0) (CONS (F (CAR L)) (F (CDR L))))
        (T (F (CAR L)))
    )
)
```

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

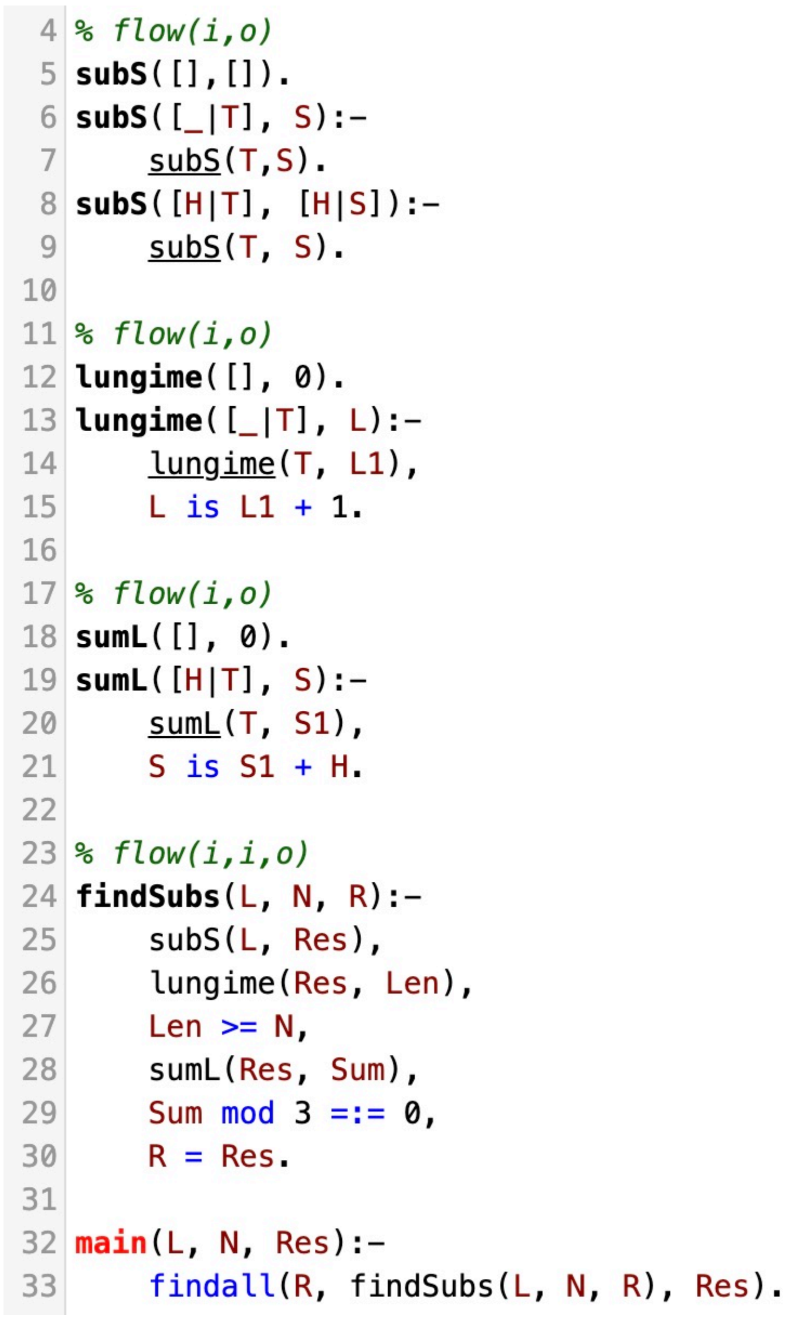
```
(DEFUN F(L)
    ((lambda (v)
        (COND
            ((NULL L) NIL)
            (> (F (CAR L)) 0) (CONS (F (CAR L)) (F (CDR L))))
            (T (F (CAR L)))
        ))(F(car L)))
    )
```

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]].



```prolog
% lcm = a*b / gdc(a,b)
%A-int, B-int, Res-int
gcd(A, 0, A).
gcd(A, B, Res):-
    B\=0,
    R is A mod B,
    gcd(B, R, Res).

%A-int, B-int, Res-int
lcm(A, B, Res):-
    gcd(A, B, R),
    I is A * B,
    Res is I // R.

lcmList([], 1).
lcmList([H|T], Res):-
    lcmList(T, R),
    lcm(H, R, Res).

getHead([H|_], H).

getLists([],[]).
getLists([H|T], [HRev|Res]):-
    is_list(H),
    getHead(H, HH),
    Sq is HH * HH,
    lcmList(H, R),
    R > Sq,
    !,
    reverse(H, HRev),
    getLists(T, Res).
getLists([H|T], [H|Res]):-
    getLists(T, Res).

%getLists([[4,1,18],7,2,3,[6,9,11,3],4,[9,4,3]], R)
```

C. Write a PROLOG program that generates the list of all subsets with at least N elements such that the value of sum of all elements from each subset is divisible with 3, from a list of integers. Write the mathematical models and flow models for the predicates used. For example, for the list L=[2,3,4] and **N**=1 ⇒ [[3],[2,4],[2,3,4]] (not necessarily in this order).



```prolog
% flow(i,o)
subS([],[]).
subS([_|T], S):-
    subS(T,S).
subS([H|T], [H|S]):-
    subS(T, S).

% flow(i,o)
lungime([], 0).
lungime([_|T], L):-
    lungime(T, L1),
    L is L1 + 1.

% flow(i,o)
sumL([], 0).
sumL([H|T], S):-
    sumL(T, S1),
    S is S1 + H.

% flow(i,i,o)
findSubs(L, N, R):-
    subS(L, Res),
    lungime(Res, Len),
    Len >= N,
    sumL(Res, Sum),
    Sum mod 3 =:= 0,
    R = Res.

main(L, N, Res):-
    findall(R, findSubs(L, N, R), Res).
```

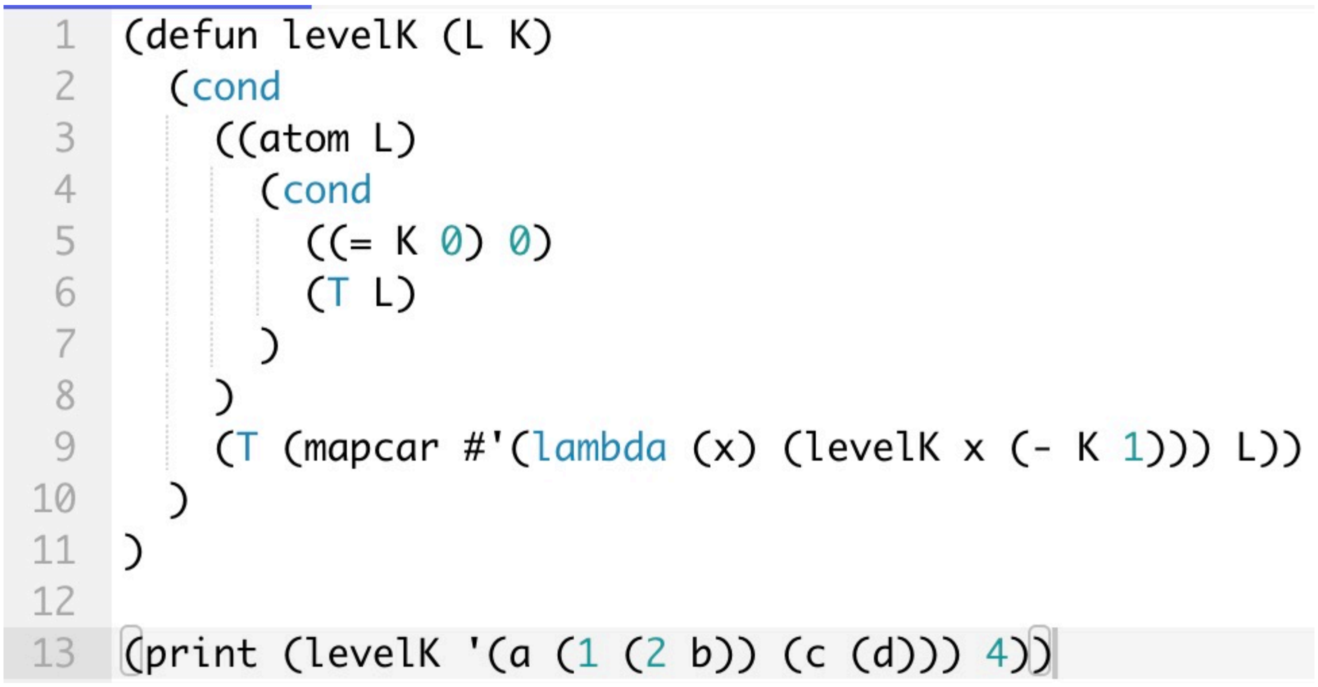
D. Given a nonlinear list, write a Lisp function to return the list with all atoms on level **k** replaced by **0**. The superficial level is assumed 1. **A MAP function shall be used.**

Example for the list (a (1 (2 b)) (c (d)))

(a) k=2 => (a (0 (2 b)) (0 (d)))

(b) k=1 => (0 (1 (2 b)) (c (d)))

(c) k=4 => the list does not change



```
(defun levelK (L K)
  (cond
    ((atom L)
      (cond
        ((= K 0) 0)
        (T L)
      )
    )
    (T (mapcar #'(lambda (x) (levelK x (- K 1))) L))
  )
)

(print (levelK '(a (1 (2 b)) (c (d))) 4))
```