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

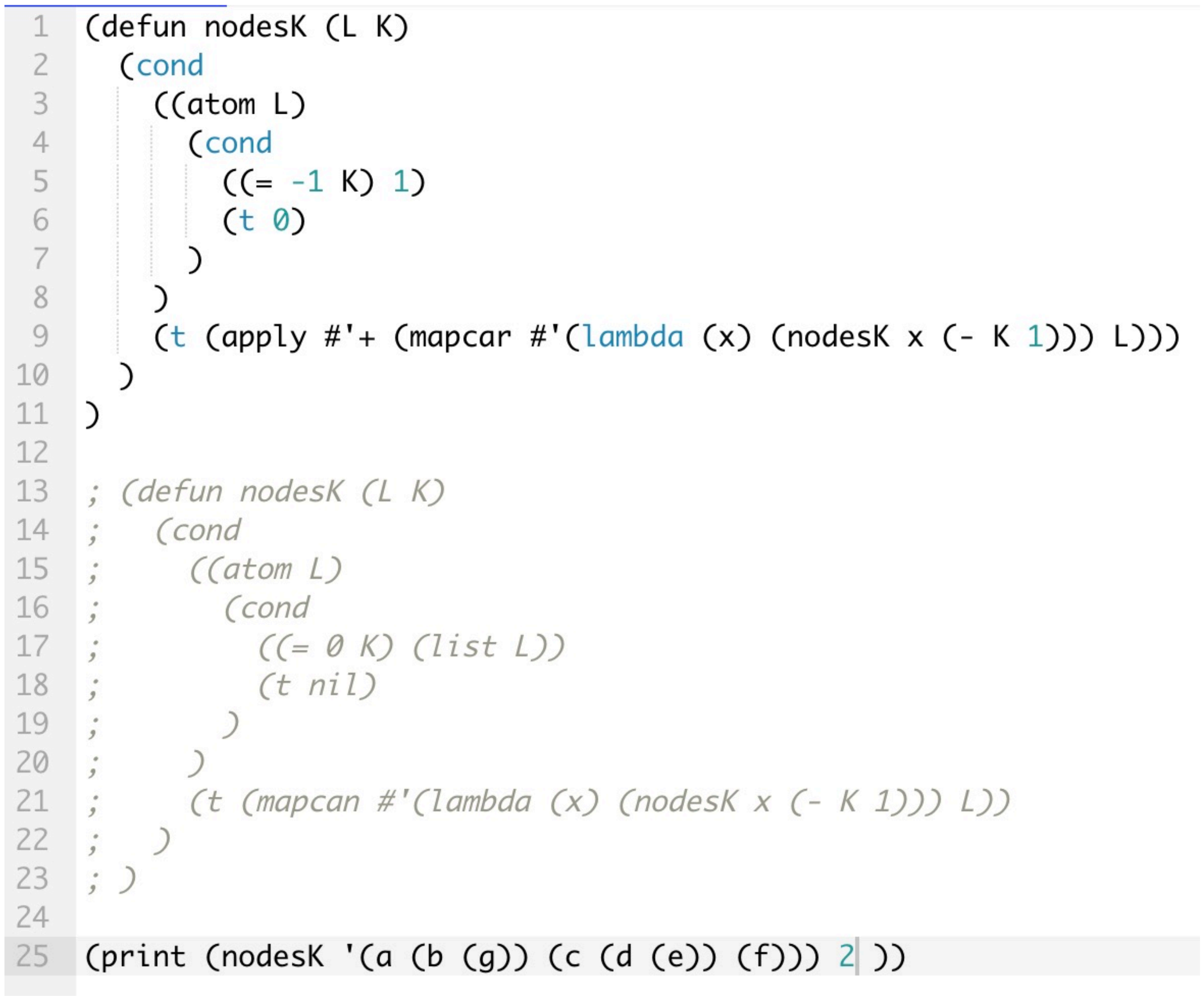
```
(DEFUN F(L)
    (COND
        ((NULL L) 0)
        (> (F (CAR L)) 2) (+ (F (CDR L)) (F(CAR L))))
        (T (+ (F (CAR L)) 1))
    )
)
```

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.

B. Given a nonlinear list composed of numbers greater or equal to 2, write a SWI-PROLOG program that replaces each nonprime number with the sum of its own proper divisors. Repeat the process until the list contains only prime numbers. **<u>For example</u>**, for the list [10, 20, 30, 40] the result will be [7, 7, 41, 7] (the initial list becomes first [7, 21, 41, 49], then [7, 10, 41, 7] and finally [7, 7, 41, 7]). Return only the final list.

C. Given a list made of integer numbers, generate in PROLOG the list of all subsets with even number of elements. Write the mathematical models and flow models for the predicates used. For example, for the list L=[2,3,4] ⇒ [[],[2,3],[2,4],[3,4]] (not necessarily in this order).

D. An n-ary tree is represented in Lisp as (node subtree1 subtree2 ...). Write a Lisp function to determine the number of nodes on level **k**. The root level is assumed zero. **A MAP function shall be used.** ***Example*** for the tree (a (b (g)) (c (d (e)) (f)))
a) k=2 => nr=3 (g d f)   **b)** k=4 => nr=0 ()



```
(defun nodesK (L K)
  (cond
    ((atom L)
      (cond
        ((= -1 K) 1)
        (t 0)
      )
    )
    (t (apply #'+ (mapcar #'(lambda (x) (nodesK x (- K 1))) L)))
  )
)

; (defun nodesK (L K)
;   (cond
;     ((atom L)
;       (cond
;         ((= 0 K) (list L))
;         (t nil)
;       )
;     )
;     (t (mapcan #'(lambda (x) (nodesK x (- K 1))) L))
;   )
; )

(print (nodesK '(a (b (g)) (c (d (e)) (f))) 2 ))
```