19:38 LTE 49

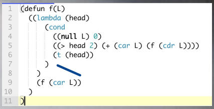
drive.google.com Done

PLF - Pop Horia

Dosare

A5. Grading: (def) – 1p; (1) – 3p (4x0.75p); (2) – 3p; (3) – 3p.

The Prolog problems *will be solved in SWI Prolog. You will explain the code, give the reasoning, predicates specification including recursive formula, flow model, meaning of all variables and parameters.* ***The Lisp problems*** *will be solved in Common Lisp. You will explain the code, give the reasoning, functions specification, meaning of all variables and parameters, the formula for recursion. The MAP problem implies writing a main and an auxiliary function. For a penalty, this may be solved without using MAP functions.*



```
(defun f(L)
  ((lambda (head)
      (cond
        ((null L) 0)
        ((> head 2) (+ (car L) (f (cdr L))))
        (t (head))
      )
    )
    (f (car L))
  )
)
```

I.

I.1 Consider the following function definition in LISP

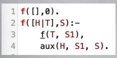
```
(DEFUN F(L)
    (COND
        ((NULL L) 0)
        ((> (F (CAR L)) 2) (+ (CAR L) (F (CDR L))))
        (T (F (CAR L)))))
```

Give a solution to avoid the double recursive call (F (CAR L)). You will not use SET, SETQ, SETF. Justify the answer.

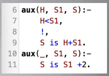
I.2 Let L be a numerical list and consider the following PROLOG definition for the predicate **f(list, integer)**, with the flow model (i, o):

```
f([], 0).
f([H|T],S):-f(T,S1),H<S1,!,S is H+S1.
f([_|T],S):-f(T,S1), S is S1+2.
```

Give a solution to avoid the recursive call f(T,S1) in both clauses without redefining the predicate. Justify the answer.



```
f([],0).
f([H|T],S):-
    f(T, S1),
    aux(H, S1, S).
```



```
aux(H, S1, S):-
    H<S1,
    !,
    S is H+S1.
aux(_, S1, S):-
    S is S1 +2.
```

I.3 The LISP function F is defined by

```
(DEFUN F(X &REST Y)
    (COND
        ((NULL Y) X)
        (T (APPEND X (MAPCAR #'CAR Y)))
    )
)
```

(12) (34 5 7) => (1 2 3 4 5 7)

What is the result of evaluating the form (APPEND (F '(1 2)) (F '(3 4) '(5 6) '(7 8)))? Justify the answer.

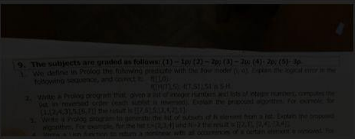
I.4 Consider the PROLOG predicates **p(integer), q(integer), r(integer)** with the flow model **(o)** and the predicate **s**.

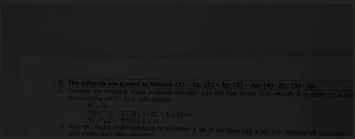
```
p(1).        q(1).        r(1).
p(2).        q(2).        r(2).
s :- !, p(X), q(Y), r(Z), write(X,Y,Z), nl.
```

Give the result of the following goal: **s.** Justify the answer.

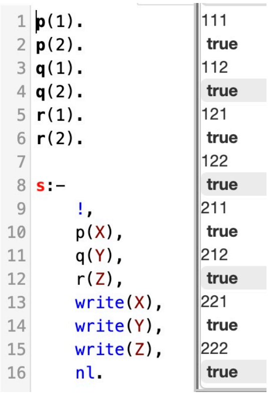
II. For a list of integer numbers, write a PROLOG program to generate the list of all subsets with at least N elements, each subset having sum of elements divisible with 3. Write the mathematical model, flow model and the meaning of all variables for each predicate used. Eg: for list L=[2,3,4] and N=1 ==> [[3],[2,4],[2,3,4]] (not necessarily in this order)

III. A nonlinear list is given. Write a LISP function to return as result the initial list in which the atoms from the level **k** from the initial list have been replaced with 0 (the superficial level is considered 1). **Use a MAP function.** Write the mathematical model and the meaning of all parameters for each function used. Eg: for list (a (1 (2 b)) (c (d))) and a) k=2 => (a (0 (2 b)) (0 (d)))





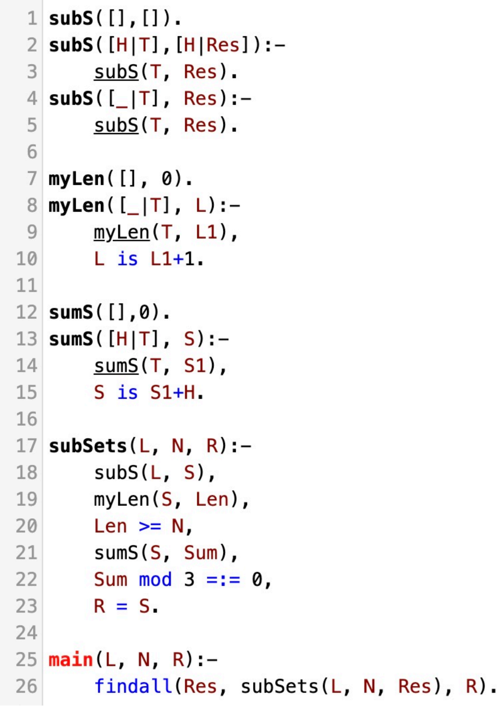
1.4



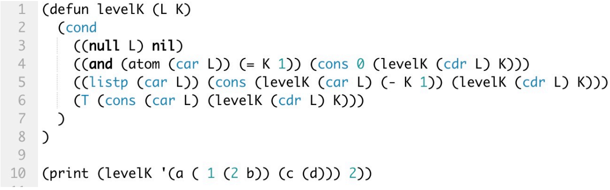
```prolog
p(1).
p(2).
q(1).
q(2).
r(1).
r(2).

s:-
    !,
    p(X),
    q(Y),
    r(Z),
    write(X),
    write(Y),
    write(Z),
    nl.
```

```
111
 true
112
 true
121
 true
122
 true
211
 true
212
 true
221
 true
222
 true
```

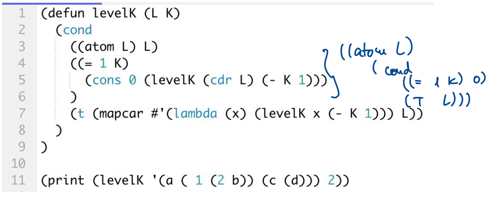
II.



```prolog
subS([],[]).
subS([H|T],[H|Res]):-
    subS(T, Res).
subS([_|T], Res):-
    subS(T, Res).

myLen([], 0).
myLen([_|T], L):-
    myLen(T, L1),
    L is L1+1.

sumS([],0).
sumS([H|T], S):-
    sumS(T, S1),
    S is S1+H.

subSets(L, N, R):-
    subS(L, S),
    myLen(S, Len),
    Len >= N,
    sumS(S, Sum),
    Sum mod 3 =:= 0,
    R = S.

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



```lisp
(defun levelK (L K)
  (cond
    ((null L) nil)
    ((and (atom (car L)) (= K 1)) (cons 0 (levelK (cdr L) K)))
    ((listp (car L)) (cons (levelK (car L) (- K 1)) (levelK (cdr L) K)))
    (T (cons (car L) (levelK (cdr L) K)))
  )
)

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



```lisp
(defun levelK (L K)
  (cond
    ((atom L) L)
    ((= 1 K)
      (cons 0 (levelK (cdr L) (- K 1)))
    )
    (t (mapcar #'(lambda (x) (levelK x (- K 1))) L))
  )
)

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

```
((atom L)
   ( cond
      ((= 1 K) 0)
      (T   L)))
```