Cusiac Andrei
Gr. 222

14-02-2022

Examen PLA
Nr. 24?

(A.) Pentru a evita apelul recursiv repetat (F (COR L)) folosim o funcție anonimă locală, apelată cu parametrul repetat

```
(DEFUN F (L)
  ((LAMBDA (X)
     (COND
       ((NULL L) 0)
       ((> (CAR L) 0)
        (COND
          ((> (CAR L) X) (CAR L))
          (T X)
        )
       )
       (T X)
     )
     (F (CDR L))
  )
)
```

Cusiac Andre.
Gr. 222

(B.)
```
%model(o,i), nedeterminist
candidat(X,[X|_]).   % X = element, [X|_] = lista
candidat(X,[_|T]):-
    candidat(X,T).
%model(i,i,o), nedeterminist
aranjamente(L,N,S,A):-   % L = lista, N = nr. elem., S = suma, A = solutia
    candidat(E,L),
    mod(E,2)=:=1,
    E+A=<S,
    aranj aux(L,N,S,A,1,E,[E]).   % nr. elem. curent, lista curenta
%model(i,i,i,o,i,i,i), N, nedeterm.
aranj aux(_,N,S,A,N,S,A):-!.
aranj aux(L,N,S,A,Lg,Sum,[H|T]):-
    Lg<N,
    candidat(E,L),
    \+ apare(E,[H|T]),
    Sum1 is Sum+E,
    Sum1=<S,
    Lg1 is Lg+1,
    aranj aux(L,N,S,A,Lg1,Sum1,[E,H|T]).
```

p. (2)

Cusiac Andrei
Gr. 222

→ model (i, i) — determinist
apare (X, [X | _]). X - elem
apare (X, [_ | T]) :- [X|_] - lista
    apare (X, T).
→ model (i, i, i, o) - nedeterminist
aranj_All (L, N, S, A) :-
    findall (P, aranjamente (L, N, S, P), A).

Folosim un predicat candidat care generează elementele unei liste.

$$candidat(l_1, \dots, l_n) = \begin{cases} 1.\ l_1, & n \neq 0 \\ 2.\ candidat(l_2, \dots, l_n) \end{cases}$$

Prin el, generăm primul element al soluției în aranjamente

aranjamente (L1 L2 ... Ln, N, S) =

= ~~1. L1, N = 1, S = L1~~

~~2. aranjamente (L2 ... Ln, N, S)~~

⇒

= aranj_aux (L1 L2 ... Ln, N, S, 1, candidat (L), [candidat (L)])

Cusiac Andrei
Gr. 222

Predicatul auxiliar aranj aux (generează
este un aranjament de sumă dată.

aranj aux (L, H, S, Lg, Sum, A) =
= n A, dacă (Sum = S și Lg = H)
n. aranj aux (L, H, S, Lg+1, Sum+e,
e ∪ A), dacă (Sum ≠ S sau H ≠ Lg) și
(Lg < k) și (e = cand. dat (L) și
e ∉ A) și (Sum + e <= S)

Predicatul apare ver. fiică apartenența unui element l_n a listă.

$$apare(X, l, l_2 .. l_n) = \begin{cases} T, \text{ dacă } X = l_1 \\ F, \text{ dacă } n = 0 \\ apare(X, l_2 .. l_n) \\ \text{altfel} \end{cases}$$

p. 4

Cusiac
Gr

(c)

Cursiac Andrei
Gr. 222

(c) $swap(l, niv, k) = \begin{cases} 0, \text{ dacă } l = atom \text{ și } niv = k \\ (l), \text{ dacă } l = atom \\ \bigcup_{i=1}^{niv} elim(l_i, niv+1, k), \text{ altfel} \end{cases}$

apel (l, k) = swap (l, 0, k)

```
(defun apel (l k)
   (swap l 0 k)
)

(defun swap (l niv k)
   (cond
      ((and (atom l) (= niv k)) 0)
      ((atom l) (l))
      (t (mapcan #'(lambda (x)
                     (swap x (+ 1 niv) k))
                 l))
   )
)
```

Cusiac Andrei
Gr. 242

Folosim o variabilă auxiliară min pt a memora nivelul actual. Când nivelul dat, întocmim cu o toți atomii de pe nivelul actual. Apoi, returnăm atomii și parcurgem recursiv listele întâlnite.