**РК 2**

**Овчинникова А. П.**

**ИУ7-65Б**

**Группа 2**

; реализовано с помощью функционала

; На всех уровнях заданного списка найти количество элементов

; принадлежащих заданному множеству.

(defun my\_find (lst set)

(defun inner (elem)

(cond

( (listp elem) (mapcan #'inner elem) )

( (my\_member elem set) (list 1) )

)

)

(reduce #'+ (mapcan #'inner lst))

)

--------------------------------------------------------------------------------

; рекусивная и нерекурсивная версия функции, кот. определяет, принадлежит ли элем.

; списку (проверяет только верхний уровень)

(defun my\_member (el lst)

(cond

( (and (null lst) (eql el Nil)) t)

( (null lst) nil )

( (equal el (car lst)) t )

( t (my\_member el (cdr lst)) )

)

)

(defun my\_member2 (el lst)

(some #'(lambda (elem)

(cond

( (equal elem el) t )

)

)

lst

)

)

--------------------------------------------------------------------------------

; рекурсивные, используют my\_member или my\_member2

; На всех уровнях заданного списка найти количество элементов, принадлежащих заданному множеству.

(defun my\_find2 (lst set)

(cond

( (null lst) 0 )

( (listp (car lst)) (+ (my\_find2 (car lst) set) (my\_find2 (cdr lst) set)) )

( (my\_member (car lst) set) (+ 1 (my\_find2 (cdr lst) set)) )

( t (+ (my\_find2 (cdr lst) set)) )

)

)

(defun inner(lst set base)

(cond

( (null lst) base )

( (listp (car lst)) (+ (inner (car lst) set 0) (inner (cdr lst) set base)) )

( (and (not (listp (car lst))) (my\_member (car lst) set)) (inner (cdr lst) set (+ base 1)) )

( (and (not (listp (car lst))) (not (my\_member (car lst) set))) (inner (cdr lst) set base) )

)

)

(defun my\_find3 (lst set)

(inner lst set 0)

)

(my\_find2 '(1) '(2 3 4))

--------------------------------------------------------------------------------

; замена k-го элемента верхнего уровня исходного списка на найденное значение

; индексация с 0

(defun rinner (lst k value res)

(cond

( (null lst) res)

( (= k 0) (nconc (nconc res (cons value Nil)) (cdr lst) ))

( t (rinner (cdr lst) (- k 1) value (nconc res (cons (car lst) Nil))) )

)

)

(defun my\_replace (lst set k)

(cond

( (< k 0) '(k меньше 0!) )

( t (rinner lst k (my\_find lst set) Nil) )

)

; можно исп. любой из приведенных выше вариантов функции my\_find

)