РК ЛИСП   
СУШИНА А ИУ7-61б

ВАРИАНТ 2

РЕКУРСИЯ

(defun in\_set(el st)

( cond ((null st) Nil)

((equal el (car st)) t)

(t (in\_set el (cdr st)))

)

)

Рекурсия. Проверяет входит ли в список.

(defun count\_in\_set(lst st res)

( cond ((null lst) res)

((listp (car lst)) ( count\_in\_set (cdr lst) st (count\_in\_set (car lst) st res)))

((in\_set (car lst) st) (count\_in\_set (cdr lst) st (+ res 1)))

(t (count\_in\_set (cdr lst) st res))

)

)

Рекурсия. Считает количество элементов входящих в множество из структурированного списка.(с доп параметром)

(defun my\_count\_in\_set(lst st)

(count\_in\_set lst st 0)

)

Рекурсия. Считает количество элементов входящих в множество из структурированного списка.(без доп параметра)

(defun myappend (lst obj)

(cond

((null lst) (cons obj nil))

((not (listp lst)) (cons lst (cons obj nil) ))

(T (cons (car lst) (myappend (cdr lst) obj)))

)

)

Рекурсия. Добавляет элемент в конец списка.

(defun replace\_k\_to\_my(lst k el res)

(cond ((null lst) (cond ((equal el nil) (cdr res))

(t (cdr (myappend res el)))))

((equal k 0) (replace\_k\_to\_my (cdr lst) (- k 1) nil (myappend res el)))

(t (replace\_k\_to\_my (cdr lst) (- k 1) el (myappend res (car lst))))

)

)

Рекурсия. Вставляет элемент на Кую позицию или в конец списка

(print (replace\_k\_to\_my '(1 2 3 4) 1 'wewe '(nil))) → (1 wewe 3 4)

(print (replace\_k\_to\_my '(1 2 3 4 wewe ) 5 'wewe '(nil))) → (1 2 3 4 WEWE WEWE)

(print (myappend '(1 2 3 4) 5)) → (1 2 3 4 5)

(defun my\_length(lst)

( cond ((null lst) 0)

((not (listp lst)) 1)

(t (+ 1 (my\_length (cdr lst)))

)

)

)

Рекурсия. Вычисляет длину списка. Для элемента возвращает 1.

(defun count\_and\_insert(lst st k)

(replace\_k\_to\_my lst k (my\_count\_in\_set lst st) `(nil))

)

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

Заменить К-ый элемент верхнего уровня исходного списка на найденное значение или дописать это значение в конец исходного списка

ФУНКЦИОНАЛЫ

(print (my\_count\_in\_set '(1 2 ( 1 5 4 (1(4)) ( 5 7 1))) '(1 2 3))) → 5

(print (in\_set 5 `(1 2 3))) → nil

(defun f\_in\_set(el sett)

(remove-if #'( lambda (x)

(not(equal x el))) sett)

)

Функционалы. Проверяет входит ли элемент в список.

(defun f\_my\_count\_in\_set (lst settt)

(reduce #'+

(mapcar #'(lambda (x)

(cond ((listp x) (f\_my\_count\_in\_set x settt))

((f\_in\_set x settt) 1)

(t 0)

)

)

lst)

)

)

Функционалы. Считает количество элементов входящих в множество из структурированного списка.

(print (f\_my\_count\_in\_set '(1 2 ( 1 5 4 (1(4)) ( 5 7 1))) '(1 2 3))) → 5

(print (f\_in\_set 1 `(1 2 3))) → (1)

(defun f\_len(lst)

(cond ((not (listp lst)) 1)

(t

(reduce #'+ (mapcar #'(lambda (x) 1) lst)))))

Функционалы. Вычисляет длину списка. Для элемента 1.

(print (f\_len '1) ) - 1

(defun rep\_k(lst l el)

(cond ((< l (f\_len lst))

( reduce #'(lambda (prev x)

( cond ((and (not ( listp prev)) (equal l 0)) (cons el (cons x nil)))

((equal (f\_len prev) l ) (myappend prev el ))

(t (myappend prev x))

))

lst

))

(t (myappend lst el)))

)

Функционалы Вставляет элемент на к позицию или в конец

(print (rep\_k '(1 2 3 5 ) 5 'el)) - (1 2 3 5 EL)

(defun c\_and\_i(lst st k)

(rep\_k lst k (f\_my\_count\_in\_set lst st))

)

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

Заменить К-ый элемент верхнего уровня исходного списка на найденное значение или дописать это значение в конец исходного списка

(print (count\_and\_insert '(1 (2 (1 a a 4) 3) 0 0 0 0) '( 1 2 a) 2 )) → (1 (2 (1 A A 4) 3) 5 0 0 0)

(print (c\_and\_i '(1 (2 (1 a a 4) 3) 0 0 0 0) '( 1 2 a) 2 )) → (1 (2 (1 A A 4) 3) 5 0 0 0)