Зейналов З. Г. ИУ7-61Б

Группа 2

Рекурсивная реализация

; функция аналогичная функции append, вставляет элемент в конец списка

(defun myappend (lst obj)

    (cond

    ((null lst) obj)

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

    )

)

; Находится ли число в множестве, результат T или nil (Рекурсия)

; Входные параметры число и множество

(defun isInRec(num set)

    (cond

    ((null set) nil)

    ((if (eql num (car set)) T nil))

    (T (isInRec num (cdr set)))

    )

)

; Поиск количества вхождений чисел из lst в множество set

 (defun find\_rec (lst set)

     (cond

     ((null lst) 0)

     ((listp lst) ( + (find\_rec (car lst) set) (find\_rec (cdr lst) set)))

     (T ( if (isInRec lst set) 1 0))

     )

 )

; Вставляет найденное количество в конец списка

(defun insertK (lst set)

    (append lst (cons (find\_rec lst set) nil))

)

Реализация с использованием функционалов.

; Находится ли число в множестве, результат T или nil (Рекурсия)

; Входные параметры число и множество

(defun isInRec(num set)

    (cond

    ((null set) nil)

    ((if (eql num (car set)) T nil))

    (T (isInRec num (cdr set)))

    )

)

; Поиск длины списка

(defun my\_length(lst)

    ( cond ((null lst) 0)

        ((not (listp lst)) 1)

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

    )

)

; Ищет кол-во вхождений эл списка в множество

(defun findn (lst set)

    (my\_length (remove nil (mapcar #'(lambda (x) (if (isIn x set) x nil)) lst)))

)

; Вставляет на к место элемент obj в список lst

(defun insertKn (lst k obj)

    (reduce

    (lambda (res cur)

        (cond

        ((= (my\_length res) k) (myappend res (cons obj nil)))

        ((listp cur) (myappend res cur))

        (T (myappend res (cons cur nil)))

        )

    ) lst :initial-value Nil)

)

; Оболочка

(defun ins (lst set k)

    (insertKn lst k (findn lst set))

)