**Лабораторная работа №6: Рекурсивные функции на Lisp (Талышева Олеся ИУ7-65Б)**

**Используя рекурсию:**

**1. Написать хвостовую рекурсивную функцию my-reverse, которая развернет верхний уровень своего списка-аргумента lst**

(defun my-reverse (lst res)

(cond ((null lst) res)

(t (my-reverse (cdr lst) (cons (car lst) res)))))

(my-reverse '(1 2 (s 4) ty 4s) Nil)

==> (4S TY (S 4) 2 1)

**2. Написать функцию, которая возвращает первый элемент списка-аргумента, который сам является непустым списком**

(defun first\_no\_empty\_elem\_lst (lst)

(cond ((null lst) Nil)

((and (listp (car lst)) (not (null (car lst)))) (car lst))

(t (first\_no\_empty\_elem\_lst (cdr lst)))))

(first\_no\_empty\_elem\_lst '(1 () 2 (s 4) ty 4s))

==> (S 4)

**3. Напишите рекурсивную функцию, которая умножает на заданное число-аргумент все числа из заданного списка-аргумента, когда**

**a) Все элементы списка - числа**

(defun multy\_num (lst num)

(cond ((null lst) Nil)

(t (cons (\* (car lst) num) (multy\_num (cdr lst) num)))))

(multy\_num '(1 4 13 5) 2)

==> (2 8 26 10)

**б) Элементы списка — любые объекты**

(defun multy\_num (lst num)

(cond ((null lst) Nil)

(t (cons (if (numberp (car lst)) (\* (car lst) num) (car lst)) (multy\_num (cdr lst) num)))))

(multy\_num '(1 () 2 (s 4) ty 4s) 2)

==> (2 NIL 4 (S 4) TY 4S)

**в) Учитываются вложенные списки**

(defun multy\_num (lst num)

(cond ((null lst) Nil)

((atom lst) lst)

(t (cons (if (numberp (car lst)) (\* (car lst) num) (multy\_num (car lst) num)) (multy\_num (cdr lst) num)))))

(multy\_num '(1 () 2 (s 4) ty 4s) 2)

==> (2 NIL 4 (S 8) TY 4S)

**4. Напишите функцию select-between, которая из списка-аргумента, содержащего только числа, выбирает только те, которые расположены между двумя указанными границами-аргументами и возвращает их в виде списка (упорядоченного по возрастанию списка чисел (+2 балла)).**

(defun insert\_help (x lst)

(cond ((null lst) (list x))

((<= x (car lst)) (cons x lst))

(t (cons (car lst) (insert\_help x (cdr lst))))))

(defun select-between (lst num1 num2 res)

(cond ((null lst) res)

((or (and (<= (car lst) num1) (>= (car lst) num2))

(and (<= (car lst) num2) (>= (car lst) num1))) (select-between (cdr lst) num1 num2 (insert\_help (car lst) res)))

(t (select-between (cdr lst) num1 num2 res))))

(select-between '(7 8 3 5 6 2 94 6 34 -2) 3 10 Nil)

==> (3 5 6 6 7 8)

**5. Написать рекурсивную версию (с именем rec-add) вычисления суммы чисел заданного списка:  
а) Одноуровневого смешанного**

(defun rec-add (lst)

(cond ((null lst) 0)

(t (+ (if (numberp (car lst)) (car lst) 0) (rec-add (cdr lst))))))

(rec-add '(1 2 s 4 ty 4s))

==> 7

**б) Структурированного**

(defun rec-add (lst)

(cond ((null lst) 0)

(t (+ (cond ((numberp (car lst)) (car lst))

((atom (car lst)) 0)

(t (rec-add (car lst)))) (rec-add (cdr lst))))))

(rec-add '(1 () 2 (s 4) ty 4s))

==> 7

**6. Написать рекурсивную версию с именем recth функции nth**

(defun recnth (n lst)

(cond ((null lst) Nil)

((= n 0) (car lst))

(t (recnth (- n 1) (cdr lst)))))

(recnth 3 '(1 (3 4 t) 4s 5 r 4))

==> 5

**7. Написать рекурсивную функцию allodd, которая возвращает t, когда все элементы списка нечётные.**

(defun allodd (lst)

(cond ((null lst) t)

((evenp (car lst)) Nil)

(t (allodd (cdr lst)))))

(allodd '(1 2 3))

==> NIL

(allodd '(1 5 3))

==> T

**8. Написать рекурсивную функцию, которая возвращает первое нечётное число из списка (структурированного), возможно создавая некоторые вспомогательные функции.**

(defun first\_odd (lst)

(cond ((null lst) Nil)

((atom (car lst)) (if (and (numberp (car lst))(oddp (car lst)))

(car lst) (first\_odd (cdr lst))))

(t (or (first\_odd (car lst)) (first\_odd (cdr lst))))))

(first\_odd '(2 4 (s 6 t5) (t 3) ((e r) 6 5))) ==> 3

**9. Используя cons-дополняемую рекурсию с одним тестом завершения, написать функцию, которая получает как аргумент список чисел, а возвращает список квадратов этих чисел в том же порядке.**

(defun squaring-list (lst)

(cond ((null lst) Nil)

(t (cons (\* (car lst) (car lst)) (squaring-list (cdr lst))))))

(squaring-list '(2 5 3 4 -6))

==> (4 25 9 16 36)

**10. Преобразовать структурированный список в одноуровневый.**

(defun perform-list (lst)

(cond ((null lst) Nil)

((atom (car lst)) (cons (car lst) (perform-list (cdr lst))))

(t (append (perform-list (car lst)) (perform-list (cdr lst))))))

(perform-list '(2 5t 3 (sd (ert 4)) () -6))

==> (2 5T 3 SD ERT 4 NIL -6)