Используя рекурсию:

## Задание 1

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

**(defun** move-to **(**lst result**)**

**(cond** **((null** lst**)** result**)**

**(**T **(**move-to **(cdr** lst**)** **(cons** **(car** lst**)** result**)))))**

**(defun** my-reverse **(**lst**)**

**(**move-to lst **()))**

## Задание 2

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

**(defun** get-first-lst**(**lst**)**

**(cond** **((null** lst**)** NIL**)**

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

**(car** lst**))**

**(**T **(**get-first-lst **(cdr** lst**)))))**

## Задание 3

Напишите рекурсивную функцию, которая умножает на заданное число-аргумент все числа из заданного списка-аргумента, когда  
a) все элементы списка --- числа,  
6) элементы списка -- любые объекты.

;numbers

**(defun** mul1**(**n lst**)**

**(cond** **((null** lst**)** NIL**)**

**(**T **(setf** **(car** lst**)** **(\*** n **(car** lst**)))**

**(**mul1 n **(cdr** lst**))))**

lst**)**

;objects

**(defun** mul2**(**n lst**)**

**(cond** **((null** lst**)** NIL**)**

**((numberp** **(car** lst**))**

**(setf** **(car** lst**)** **(\*** n **(car** lst**)))**

**(**mul2 n **(cdr** lst**)))**

**(**T **(**mul2 n **(cdr** lst**))))**

lst**)**

## Задание 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** s-b-r**(**lst a b res**)**

**(cond** **((null** lst**)** res**)**

**(((lambda** **(**el**)**

**(if** **(<** a b**)**

**(<=** a el b**)**

**(<=** b el a**)))**

**(car** lst**))**

**(**s-b-r **(cdr** lst**)** a b **(cons** **(car** lst**)** res**)))**

**(**T **(**s-b-r **(cdr** lst**)** a b res**))))**

**(defun** select-between**(**lst a b**)**

**(**sort **(**s-b-r lst a b **())** #'<**))**

## Задание 5

Написать функцию, которая выбирает из заданного списка только те числа, которые больше 1 и меньше 10. (Вариант: между двумя заданными границами.)

**(defun** between-1-10**(**lst**)**

**(**select-between lst 1 10**))**

## Задание 6

Написать рекурсивную версию (с именем rec-add) вычисления суммы чисел заданного списка:   
а) одноуровнего смешанного,  
б) структурированного.

**(defun** rec-add1-r**(**lst res**)**

**(cond** **((null** lst**)** res**)**

**((numberp** **(car** lst**))**

**(**rec-add1-r **(cdr** lst**)** **(+** res **(car** lst**))))**

**(**T **(**rec-add1-r **(cdr** lst**)** res**))))**

**(defun** rec-add1**(**lst**)**

**(**rec-add1-r lst 0**))**

**(defun** rec-add2-r**(**lst res**)**

**(cond** **((null** lst**)** res**)**

**((numberp** **(car** lst**))**

**(**rec-add2-r **(cdr** lst**)** **(+** res **(car** lst**))))**

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

**(**rec-add2-r **(cdr** lst**)**

**(**rec-add2-r **(car** lst**)** res**)))))**

**(defun** rec-add2**(**lst**)**

**(**rec-add2-r lst 0**))**

## Задание 7

Написать рекурсивную версию с именем recnth функции nth.

**(defun** recnth**(**n lst**)**

**(cond** **((=** n 0**)** **(car** lst**))**

**(**T **(**recnth **(-** n 1**)** **(cdr** lst**)))))**

## Задание 8

Написать рекурсивную функцию allodd, которая возвращает t когда все элементы списка нечетные.

**(defun** my-oddp**(**x**)**

**(if** **(=** **(rem** x 2**)** 1**)**

T**))**

**(defun** allodd**(**lst**)**

**(cond** **((null** lst**)** T**)**

**((**my-oddp **(car** lst**))** **(**allodd **(cdr** lst**)))**

**(**T NIL**)))**

## Задание 9

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

**(defun** get-odd**(**lst**)**

;(print lst)

**(cond** **((null** lst**)** NIL**)**

**((and** **(numberp** **(car** lst**))** **(**my-oddp **(car** lst**)))**

**(car** lst**))**

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

**(or** **(**get-odd **(car** lst**))** **(**get-odd **(cdr** lst**))))**

**(**T **(**get-odd **(cdr** lst**)))))**

## Задание 10

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** get-squares-r**(**lst res**)**

**(cond** **((null** lst**)** res**)**

**(**T **(**get-squares-r **(cdr** lst**)**

**(cons** **(\*** **(car** lst**)** **(car** lst**))** res**)))))**

**(defun** get-squares**(**lst**)**

**(**my-reverse **(**get-squares-r lst NIL**)))**