## Задание 1

Написать функцию, которая по своему списку-аргументу lst определяет является ли он палиндромом (то есть равны ли lst и (reverse lst)).

**(defun** is-palindrom **(**lst**)**

**(equal** lst **(reverse** 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 **()))**

**(defun** is-palindrom2 **(**lst**)**

**(equal** lst **(**my-reverse lst**)))**

## Задание 2

Написать предикат set-equal, который возвращает t, если два его множества-аргумента содержат одни и те же элементы, порядок которых не имеет значения.

**(defun** set1-in-set2-r **(**set1 set2 res**)**

**(cond** **((null** set1**)** T**)**

**((member** **(car** set1**)** set2**)**

**(**set1-in-set2-r **(cdr** set1**)** set2 res**))**

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

**(defun** set1-in-set2 **(**set1 set2**)**

**(**set1-in-set2-r set1 set2 T**))**

**(defun** set-equal**(**set1 set2**)**

**(and** **(**set1-in-set2 set1 set2**)**

**(**set1-in-set2 set2 set1**)))**

## Задание 3

Напишите свои необходимые функции, которые обрабатывают таблицу из 4-х точечных пар:

(страна . столица), и возвращают по стране - столицу, а по столице — страну.

**(defun** get-cap **(**tab coun**)**

**(cond** **((null** tab**)** NIL**)**

**((eql** **(caar** tab**)** coun**)** **(cdar** tab**))**

**(**T **(**get-cap **(cdr** tab**)** coun**))))**

**(defun** get-coun **(**tab cap**)**

**(cond** **((null** tab**)** NIL**)**

**((eql** **(cdar** tab**)** cap**)** **(caar** tab**))**

**(**T **(**get-coun **(cdr** tab**)** cap**))))**

## Задание 4

Напишите функцию swap-first-last, которая переставляет в списке-аргументе первый и последний элементы.

**(defun** without-last-r**(**lst res**)**

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

**(**T **(**without-last-r **(cdr** lst**)**

**(append** res **`(**,**(car** lst**)))))))**

**(defun** without-last**(**lst**)**

**(**without-last-r lst **()))**

**(defun** swap-first-last**(**lst**)**

**(cond** **((or** **(null** lst**)** **(null** **(cdr** lst**)))** lst**)**

**(**T **(append** **(last** lst**)**

**(**without-last **(cdr** lst**))**

**`(**,**(car** lst**))))))**

## Задание 5

Напишите функцию swap-two-ellement, которая переставляет в списке- аргументе два указанных своими порядковыми номерами элемента в этом списке.

**(defun** swap-two-ellement**(**lst n1 n2**)**

**(let** **((**tmp **(nth** n2 lst**)))**

**(setf** **(nth** n2 lst**)** **(nth** n1 lst**))**

**(setf** **(nth** n1 lst**)** tmp**)**

lst**))**

## Задание 6

Напишите две функции, swap-to-left и swap-to-right, которые производят одну круговую перестановку в списке-аргументе влево и вправо, соответственно.

**(defun** move-left**(**lst tmp**)**

**(cond** **((null** **(cdr** lst**))** **(setf** **(car** lst**)** tmp**))**

**(**T **(setf** **(car** lst**)** **(**second lst**))**

**(**move-left **(cdr** lst**)** tmp**))))**

**(defun** swap-to-left**(**lst**)**

**(cond** **((or** **(null** lst**)** **(null** **(cdr** lst**)))** lst**)**

**(**T **(let** **((**tmp **(**first lst**)))**

**(**move-left lst tmp**)**

lst**))))**

**(defun** move-right**(**lst tmp2**)**

**(cond** **((null** lst**)** **nil)**

**(**T **(let** **((**tmp3 **(car** lst**)))**

**(setf** **(car** lst**)** tmp2**)**

**(**move-right **(cdr** lst**)** tmp3**)))))**

**(defun** swap-to-right**(**lst**)**

**(cond** **((or** **(null** lst**)** **(null** **(cdr** lst**)))** lst**)**

**(**T **(let** **((**tmp **(car** **(last** lst**))))**

**(**move-right lst **(car** lst**))**

**(setf** **(car** lst**)** tmp**)**

lst**))))**

## Задание 7

Напишите функцию, которая добавляет к множеству двухэлементных списков новый двухэлементный список, если его там нет.

**(defun** add-to-set**(**set1 el**)**

**(if** **(member** el set1 **:test** #'equal**)**

set1

**(nconc** set1 **(list** el**))))**

## Задание 8

Напишите функцию, которая умножает на заданное число-аргумент первый числовой элемент списка из заданного 3-х элементного списка-аргумента, когда

а) все элементы списка – числа,

б) элементы списка – любые объекты.

;numbers

**(defun** f81**(**n lst**)**

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

lst**)**

;objects

**(defun** f82**(**n lst**)**

**(if** **(member** T **(mapcar** #'numberp lst**))**

**(let** **((**i **(-** **(length** lst**)(length** **(member** T **(mapcar** #'numberp lst**))))))**

**(setf** **(nth** i lst**)** **(\*** **(nth** i lst**)** n**))**

lst**)))**

## Задание 9

Напишите функцию, select-between, которая из списка-аргумента из 5 чисел 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 **(append** res **`(**,**(car** lst**)))))**

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

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

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