Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 8

По дисциплине «Функциональное и логическое программирование»

Студент: Тимонин А. С.

Группа ИУ7-62б

Преподаватель Толпинская Н. Б.

Москва.
2020 г.

Практическая часть

Задание 1.

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

```
(setq lst (list 1 2 3 4 3 2 1))

(defun isPalHelper (lst lstq size)
    (if (not(or (null (car lst)) (null (car lstq))))
        (and (equal (car lst) (car lstq))
            (isPalHelper (cdr lst) (cdr lstq) (- size 1)))
        T
    )
)
; lst - список
; lstq - рекурсивный список lst
; size - длина списка lst
; isPalHelper - каждый раз сравнивает начало и конец
списка, и запускает сравнение рекурсивно

(defun isPalindrom (lst)
    (isPalHelper lst (reverse lst) (length lst))
)
; lst - список
; isPalindrom - функция-обертка, которая запускает функцию
проверку на палиндром
; (isPalindrom (list 1 2 1)) -> T
; (isPalindrom (list 1 2 3 2)) -> NIL
```

Задание 4.

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

```
(defun swap-first-last (lst)
    (setq lastEl (last lst))
    (setq firstEl (list (car lst)))
    (if (not (null (cdr lst)))
        (append
            lastEl
            (cdr (butlast lst))
```

```
            firstEl
        )
        (if (null (car lst))
            Nil
            (car lst)
        )
    )
)
; lst – список
; swap-first-last – меняет местами первый и последний
элементы
; (swap-first-last (list 1 2 3 4)) -> (4 2 3 1)
; (swap-first-last (list 1)) -> 1
; (swap-first-last ()) -> NIL
; (swap-first-last ()) -> NIL
```

Задание 5.

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

A. все элементы списка --- числа,

B. элементы списка -- любые объекты.

```
(defun swap-two-element (lst f s)
    (let
        (
            (var (nth f lst))
        )
        (setf (nth f lst) (nth s lst))
        (setf (nth s lst) var)
    )lst
)
; lst – список
; f, s – первый и второй индексы которые нужно поменять
местами
; (swap-two-element (list 1 2 3 4 5 6 7 8) 0 3) -> (4 2 3 1
5 6 7 8)
```

Задание 6.

Напишите две функции, swap-to-left и swap-to-right, которые производят круговую перестановку в списке-аргументе влево и вправо соответственно (На k позиций)

```
(defun swapToleft (lst)
    (
        append (cdr lst) (cons (first lst) nil)
    )
)
; lst – список
; swapToleft – сдвигает значения списка влево

(defun swap-to-left (lst k)
    (cond
        ( (< k 1) lst)
        ( T (swap-to-left (swapToleft lst) (- k 1)))
    )
)
; lst – список
; k – на сколько позиций нужно сдвинуть
; swap-to-left – функция-оболочка, запускающая swapToleft



(defun swapToright (lst)
    (
        append (last lst) (butlast lst)
    )
)
; lst – список
; swapToright – сдвигает значения списка вправо

(defun swap-to-right (lst k)
    (cond
        ( (< k 1) lst)
        ( T (swap-to-right (swapToright lst) (- k 1)))
    )
)
; lst – список
; k – на сколько позиций нужно сдвинуть
; swap-to-right – функция-оболочка, запускающая swapToright
```

Задание 7.

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

а) все элементы списка --- числа,

б) элементы списка --- любые объекты.

```
; а) все элементы списка --- числа
(defun multNum1 (lst num)
    (mapcar #'(lambda (x) (* x num))
        lst
    )
)
; lst – список, num – множитель
; (multNum1 (list 1 2 3) 10) -> (10 20 30)

(defun multNum2 (lst number)
    (cons
        (if (not (null lst))
            (* (car lst) number)
        )

        (cond
            ((< (length (cdr lst)) 1) Nil)
            (t (multNum (cdr lst) number))
        )
    )
)
; lst – список, number – множитель
; (multNum2 (list 1 2 3) 10) -> (10 20 30)

; б) элементы списка -- любые объекты.
(defun multAll (lst num)
    (mapcar
        #'(lambda (x)
                (cond
                    ((numberp x) (* x num))
                    ((listp x) (multAll x num))
                    (t x)
                )
          ) lst
    )
)
; lst – список, num – множитель
; (multAll (list 1 2 'a (list 5 'b 6)) 10) -> (10 20 A (50
B 60))
```

Задание 8.

Напишите функцию, select-between, которая из списка-аргумента, содержаещего 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 select-between (lst left right)
    (remove-if
        #'(lambda (x) (or (< x left) (> x right)))
        lst
    )
)
; lst - список
; left, right - левая и правая границы (числа)
; (select-between (list 1 2 3 4 5 6 7 8) 3 5) -> (3 4 5)
```