

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

---

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 8

Дисциплина	Функциональное и логическое программирование.
Тема	Работа с функционалами
Студент	Куприй А. А.
Группа	ИУ7-63Б
Преподаватель	Толпинская Н.Б., Строганов Ю.В.

Москва, 2020 г.

## 1 Практическая часть

### 1.1 Задание №1

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

```
1 (defun is_palindrom (lst)
2   (equal lst (reverse lst))
3 )
```

### 1.2 задание №2

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

```
1 (defun swap-first-last (lst)
2   (cond
3     ((null lst) Nil)
4     ((eql (length lst) 1) lst)
5     (T
6       (append
7         (last lst)
8         (mapcar #'(lambda (el _) el) (cdr lst) (cddr lst))
9         (list (first lst))
10      )
11    )
12  )
13 )
```

```
1 (swap-first-last ()) ;;; Nil
2 (swap-first-last '(1)) ;;; (1)
3 (swap-first-last '(1 2)) ;;; (2 1)
4 (swap-first-last '(1 2 3 4 5)) ;;; (5 2 3 4 1)
```

### 1.3 Задание №3

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

```
1 (defun swap-two-elements (a b lst)
```

```

2      (cond
3        ((eql a b) lst)
4        ((> a b) (swap-two-elements b a lst))
5        (T
6          (append
7            (subseq lst 0 a)
8            (list (nth b lst))
9            (subseq lst (+ a 1) b)
10           (list (nth a lst))
11           (subseq lst (+ b 1))
12         )
13       )
14     )
15   )

```

```

1 (swap-two-elements '2 '4 '(1 2 3 4 5)) ;;; (1 2 5 4 3)
2 (swap-two-elements '1 '4 '(1 2 3 4 5)) ;;; (1 5 3 4 2)
3 (swap-two-elements '3 '2 '(1 2 3 4 5)) ;;; (1 2 4 3 5)

```

## 1.4 Задание №4

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

```

1 (defun swap-to-left-one (lst)
2   (append (mapcar #'(lambda (el _) el) (cdr lst) lst) (list (car lst)))
3 )
4
5 (defun swap-to-left (lst k)
6   (cond
7     ((<= k 0) lst)
8     (T (swap-to-left (swap-to-left-one lst) (- k 1)))
9   )
10 )
11
12 (defun swap-to-right-one (lst)
13   (append (last lst) (mapcar #'(lambda (el _) el) lst (cdr lst)))
14 )
15
16 (defun swap-to-right (lst k)
17   (cond
18     ((<= k 0) lst)
19     (T (swap-to-right (swap-to-right-one lst) (- k 1)))
20   )

```

```

1 (swap-to-left '(1 2 3 4 5) 2) ;;; (3 4 5 1 2)
2 (swap-to-left '(1 2 3 4 5) 5) ;;; (1 2 3 4 5)
3
4 (swap-to-right '(1 2 3 4 5) 2) ;;; (4 5 1 2 3)
5 (swap-to-right '(1 2 3 4 5) 5) ;;; (1 2 3 4 5)

```

## 1.5 Задание №5

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.

На вход подается список и число.

С помощью функционала `mapcar`, производится работа с каждым аргументом списка: проверяется на число (если необходимо), выполняется умножение на заданный аргумент.

```

1 (defun multiplication_numbers (lst k)
2   (mapcar #'(lambda (x) (* x k)) lst)
3 )
4
5 (defun multiplication_all (lst k)
6   (mapcar #'(lambda (x) (if (numberp x) (* x k) x)) lst)
7 )

```

## 1.6 Задание №6

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).

```

1 (defun found_between (a b lst)
2   (defun found (lst1 lst2)
3     (if (numberp lst1)
4         (append(if(and(< a lst1)(< lst1 b))(list lst1) Nil)

```

```
5          (if (and (< a lst2) (< lst2 b)) (list lst2) Nil))
6      (append lst1
7          (if (and (< a lst2) (< lst2 b)) (list lst2) Nil))
8      )
9  )
10 (reduce #'found lst)
11 )
```