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»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6
по курсу «Функциональное и логическое программирование»

Студент Маслова Марина Дмитриевна

Группа ИУ7-63Б

Оценка (баллы) _____

Преподаватель Толпинская Наталья Борисовна

Преподаватель Строганов Юрий Владимирович

2022 г.

1 Практические задания

1.1 Задание №1

Напишите функцию, которая уменьшает на 10 все числа из списка-аргумента этой фукнции.

```
(defun minus10 (lst)
    (mapcar #'(lambda (x)
                  (cond ((numberp x) (- x 10))
                        ((listp x) (minus10 x))
                        (T x))) lst))
```

1.2 Задание №2

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

1. все элементы списка — числа,

```
(defun mul-num (lst num)
    (mapcar #'(lambda (x) (* x num)) lst))
```

2. элементы списка — любые объекты.

```
(defun mul-num (lst num)
    (mapcar #'(lambda (x)
                  (cond ((numberp x) (* x num))
                        ((listp x) (mul-num x num))
                        (T x))) lst))
```

1.3 Задание №3

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

```
(defun my-reverse (lst)
    (reduce #'(lambda (x y) (cons y x)) (cons nil lst)))

(defun is-palindrome (lst)
    (equal lst (my-reverse lst)))
```

1.4 Задание №4

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

```
(defun is-in (el st)
    (reduce #'(lambda (x y) (or x y))
            (mapcar #'(lambda (x) (equal x el)) st) :initial-value Nil))

(defun is-subset (st1 st2)
    (reduce #'(lambda (x y) (and x y))
            (mapcar #'(lambda (x) (is-in x st2)) st1) :initial-value T))

(defun set-equal (st1 st2)
    (and (is-subset st1 st2) (is-subset st2 st1)))
```

1.5 Задание №5

Написать функцию, которая получает как аргумент список чисел, а возвращает список квадратов этих чисел в том же порядке.

```
(defun squares (lst)
    (mapcar #'(lambda (x) (* x x) lst)))
```

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 возращает их в виде списка (упорядоченного по возрастанию списка чисел).

```
(defun sort-insert (el lst)
    (cond ((null lst) (list el))
          ((< el (car lst)) (cons el lst))
          (T (cons (car lst) (sort-insert el (cdr lst))))))

(defun select-between (lst begin end)
    (reduce #'(lambda (res x)
                (cond ((< begin x end) (sort-insert x res))
                      (T res))) lst :initial-value Nil))
```

1.7 Задание №7

Написать функцию, вычисляющую декартово произведение двух своих списков-аргументов.

```
(defun cartesian (lst1 lst2)
    (mapcan #'(lambda (el1)
                    (mapcar #'(lambda (el2) (list el1 el2)) lst2)) lst1))
```

1.8 Задание №8

Почему так реализовано reduce, в чем причина?

```
(reduce #'+ '(0)) -> 0
(reduce #'+ ()) -> 0
```

Функция проверяет список-аргумент. Если он содержит только один элемент и initial-value не задано, то возвращается элемент и функция не вызывается. Если список-аргумент пустой и initial-value не задано, то функция вызывается с нулевым количеством параметров.

1.9 Задание №9

Пусть list-of-list список, состоящий из списков. Написать фукнцию, которая вычисляет сумму длин всех элементов list-of-list, т. е., например, для аргумента ((1 2) (3 4)) -> 4.

```
(defun get-length (lst)
    (reduce #'(lambda (cur-len x)
                    (cond ((listp x) (+ cur-len (get-length x)))
                          (T (+ cur-len 1)))) lst :initial-value 0))
```