**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 №9 по дисциплине «Функциональное и логическое программирование»

**Тема** Использование функционалов и рекурсии

**Студент** Романов А.В.

**Группа** ИУ7-63Б

**Оценка (баллы)** _____

**Преподаватель** Толпинская Н.Б., Строганов Ю. В.

Москва — 2021 г.

# Задание 1

## Постановка задачи

Написать функцию, которая выбирает из заданного списка только ты числа, которые больше 1 и меньше 10.

## Решение

```
(defun select-between (lst)
  (reduce
    #'(lambda (acc el)
       (if (and (> el 1) (< el 10))
         (append acc (cons el Nil))
         acc))
  lst :initial-value ()))
```

# Задание №2

## Постановка задачи

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

## Решение

```
(defun cartesian-prod (lst1 lst2)
  (mapcan
    #'(lambda (x1)
       (mapcar
         #'(lambda (x2) (cons x1 x2))
       lst2))
  lst1))
```

# Задание №3

## Постановка задачи

Почему так реализовано `reduce`, в чем причина? `(reduce #'+ ()) -> 0`

## Решение

Функция `+` — функционал, который при 0 количестве аргументов возвращает значение 0. Если подать на вход `reduce` функцию, которая не может обработать 0 аргументов, то вызов

`reduce` с пустым списком в качестве второго аргумента вернет ошибку (`invalid number of arguments: 0`). При этом, если подано более одного аргумента, то `reduce` выполняет действия:

1. сохраняет первый элемент списка в область памяти;
2. для всех остальных элементов списка выполняет переданную в качестве первого аргумента функцию, подавая на вход 2 аргумента и сохраняя результат в `acc`.

# Задание №4

## Постановка задачи

Пусть `list-of-lists` — список, состоящий из списков. Написать функцию, которая вычисляет сумму длин всех элементов `list-of-lists`, то есть, например, для аргумента `((1 2) (3 4)) -> 4`

## Решение

```
(defun sum-len-of-list (lst)
  (reduce #'(lambda (acc x) (+ acc (length x))) lst :initial-value 0))
```

# Задание №5

Используя рекурсию, написать функцию, которая по исходному списку стоит список квадратов чисел смешанного структурированного списка

## Постановка задачи

## Решение

```
(defun sqr (x)
  (* x x))

(defun list-squares (lst)
  (defun wrapper (lst acc)
    (if (null lst) acc
      (cond ((listp (car lst))
        (append acc
            (wrapper (cdr lst) (wrapper (car lst) ()))))
         ((numberp (car lst))
            (append acc (wrapper (cdr lst) (cons (sqr (car lst)) Nil))))
        (t acc))))
  (wrapper lst ()))
```

# Контрольные вопросы

**Вопрос 1.** Классификация рекурсивных функций

**Ответ. Рекурсия** — ссылка на описываемый объект во время его описания.

Классификация рекурсивных функций:

- простая (рекурсивный вызов — единственный);
- второго порядка (несколько рекурсивных вызовов);
- взаимная рекурсия (используются несколько рекурсивных функций, которые могут друг друга вызывать).
- хвостовая рекурсия (при очередном вызове рекурсивной функции все действия до входа выполнены, а при выходе ничего более делать не приходится);
- дополняемая рекурсия (результат рекурсии используется, как аргумент некоторой другой функции (которую называют *дополняемой функцией*); частный случай — `cons`-дополняемая рекурсия).