**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 (ИУ7)**

# ОТЧЕТ

**по лабораторной работе № 9**

**Название**: Использование функционалов и рекурсии.

**Дисциплина:** Функциональное и логическое программирование

Студент ИУ7-63Б _____ В.П. Федоров
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Н.Б. Толпинская
(Подпись, дата) (И.О. Фамилия)

Москва, 2021

**Задача №4.**
*Написать функцию, которая выбирает из заданного списка только те числа, которые больше 1 и меньше 10.*
*(вариант: между двумя заданными границами).*

```
(defun get-nums(lst)
	(cond
		((null (cdr lst)) nil)
		((and (> (car lst) 1) (< (car lst) 10)) (cons (car lst) (get-nums (cdr lst))))
		(t (get-nums (cdr lst)))))
```

```
(defun get-nums(lst boardA boardB)
	(cond
		((> boardA boardB) (get-nums lst boardB boardA))
		((null (cdr lst)) nil)
		((and (> (car lst) boardA) (< (car lst) boardB)) (cons (car lst) (get-nums (cdr
lst) boardA boardB)))
		(t (get-nums (cdr lst) boardA boardB))))
```

**Задача №5.**
*Написать функцию, вычисляющую декартово произведение двух списков-аргументов.*

```
(defun decart-one-element(x lst)
	(cond
		((null lst) nil)
		(t (cons (cons x (car lst)) (decart-one-element x (cdr lst))))))

(defun decart(lstA lstB)
	(cond
		((null lstA) nil)
		(t (append (decart-one-element (car lstA) lstB)
                                                (decart (cdr lstA) lstB)))))
```

**Задача №6.**
*Почему так реализовано reduce, в чем причина?*
*(reduce #'+ ()) -> 0*

Потому что список, к которому применяется reduce, является пустым, результатом является начальное значение, которое по умолчанию равно 0.

**Задача №7.**
*Пусть list-of-list список, состоящий из списков. Написать функцию, которая вычисляет сумму длин всех элементов list-of-list.*

```
(defun list-sum-len(lst)
        (cond
                ((null (cdr lst)) (length (car lst)))
                (t (+ (length (car lst)) (list-sum-len (cdr lst))))))
```

**Дополнительная задача.**
Используя рекурсию, написать функцию, которая по исходному списку строит список из квадратов чисел смешанного структурированного списка.

```
(defun get-squares(lst)
        (cond
                ((null lst) nil)
                ((numberp (car lst)) (cons (* (car lst) (car lst)) (get-squares (cdr lst))))
                ((listp (car lst)) (append (get-squares (car lst)) (get-squares (cdr lst))))
                (t (get-squares (cdr lst)))))
```

**Вопросы.**

1. *Классификация рекурсивных функций:*

- Простые - один рекурсивный вызов в функции.
- Рекурсия первого порядка - несколько рекурсивных вызовов в функции.
- Взаимная рекурсия - несколько рекурсивных функций могут вызывать друг друга.
- Хвостовая рекурсия - последняя операция перед возвратом из функции - рекурсивный вызов.
- Дополняемая рекурсия