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, искусственный интеллект и системы управления**»

КАФЕДРА «**Программное обеспечение ЭВМ и информационные технологии**»

Отчет по лабораторной работе №6.1

по курсу «Функциональное и логическое программирование»

«Рекурсивные функции»

Студент группы **ИУ7-64Б** _____ **Д.С. Чепиго**

(Подпись, дата) (И.О. Фамилия)

Преподаватели _____ **Н.Б. Толпинская**

(Подпись, дата) (И.О. Фамилия)

_____ **Ю.В. Строганов**

(Подпись, дата) (И.О. Фамилия)

***2023** г.*

Содержание

1 Практические задания

Используя функционалы:

1.1 Задание №1

Написать хвостовую рекурсивную функцию my-reverse, которая развернет верхний уровень своего списка-аргумента lst.

```
(defun my-reverse (lst &optional (buf-lst NIL))
    (cond ((not lst) buf-lst)
          (T (my-reverse (cdr lst) (cons (car lst) buf-lst)))))
```

1.2 Задание №2

Написать функцию, которая возвращает первый элемент списка-аргумента, который сам является непустым списком.

```
(defun not-null-list (lst)
    (cond ((not lst) NIL)
        ((and (listp (car lst))
            (not (not (caar lst)))) (car lst))
        (T (not-null-list (cdr lst)))))
```

1.3 Задание №3

Напишите рекурсивную функцию, которая умножает на заданное число-аргумент все числа из заданного списка-аргумента, когда: все элементы списка – числа, элементы списка – любые объекты

```
(defun append-elem (lst elem &optional (before-lst Nil))
    (cond ((and (not lst)
                (not before-lst)) (cons elem NIL))
          ((not lst)
                (my-reverse (cons elem before-lst)))
          (T
```

```
                (append-elem (cdr lst) elem
                             (cons (car lst) before-lst)))))

(defun all-numbers (num lst &optional (res-lst NIL))
     (cond ((not lst) res-lst)
           (T (all-numbers num (cdr lst)
                (append-elem res-lst (* (car lst) num))))))

(defun not-all-numbers (num lst &optional (res-lst NIL))
     (cond ((not lst) res-lst)
           ((numberp (car lst))
                (not-all-numbers num (cdr lst)
                    (append-elem res-lst (* num (car lst)))))
           ((listp (car lst))
                (not-all-numbers num (cdr lst)
                    (append-elem res-lst
                             (not-all-numbers num (car lst)))))
           (T   (not-all-numbers num (cdr lst)
                    (append-elem res-lst (car lst))))))
```

1.4 Задание №4

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

```
(defun select-between (num1 num2 lst &optional (res-lst NIL))
     (cond  ((not lst) res-lst)
            ((< num1 (car lst) num2)
                (select-between num1 num2 (cdr lst)
                (append-elem res-lst (car lst))))
            (T (select-between num1 num2 (cdr lst) res-lst))))
```

1.5 Задание №5

Написать рекурсивную версию (с именем rec-add) вычисления суммы чисел заданного списка: одноуровнего смешанного, структурированного.

```
(defun rec-add-a (lst &optional (sum 0))
    (cond ((not lst)
                sum)
        ((numberp (car lst))
                (rec-add-a (cdr lst) (+ sum (car lst))))
        (T
            (rec-add-a (cdr lst) sum))))

(print (rec-add-a '(1 v 4 0)))

(defun rec-add-b (lst &optional (sum 0))
    (cond ((not lst)
                sum)
        ((numberp (car lst))
                (rec-add-b (cdr lst) (+ sum (car lst))))
        ((listp (car lst))
                (rec-add-b (cdr lst)
                        (+ sum (rec-add-b (car lst)))))
        (T
            (rec-add-b (cdr lst) sum))))
```