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»

на тему: «Рекурсивные функции»

Студент ИУ7-61Б _____ Савинова М. Г.
(Группа) (Подпись, дата) (Фамилия И. О.)

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

Преподаватель _____ Строганов Ю.В.
(Подпись, дата) (Фамилия И. О.)

2024 г.

1 Практические задания

1.1 Задание 1

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

```
(defun my_reverse(lst res)
      (cond ((null lst) res)
            (t (my_reverse (cdr lst) (cons (car lst) res)))
      )
)
```

1.2 Задание 2

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

```
(defun non-empty(lst)
    (cond ((null lst) Nil)
          ((listp (car lst)) (car lst))
          (t (non-empty (cdr lst)))
    )
)
```

1.3 Задание 3

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

```
(defun fit(lst)
      (cond ((null lst) Nil)
            ((listp (car lst)) (fit (car lst)))
            ((< 1 (car lst) 10) (cons (car lst) (fit (cdr lst))))
            (t (fit (cdr lst)))
      )
)
```

1.4 Задание 4

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

— все элементы списка — числа,

— элементы списка — любые объекты.

```
(defun mul (lst n)
    (cond ((null lst) Nil)
          ((listp (car lst)) (mul (car lst) n))
          ((numberp (car lst)) (cons (* (car lst) n) (mul (cdr
             lst) n)))
          (t (cons (car lst) (mul (cdr lst) n)))
    )
)
```

1.5 Задание 5

Написать функцию, select-between, которая из списка-аргумента, содержащего только числа, выбирает только те, которые расположены между двумя указанными числами — границами - аргумента и возвращает их в виде списка (упорядоченного по возрастанию).

```
(defun fit(lst a b)
      (cond ((null lst) Nil)
            ((listp (car lst)) (fit (car lst) a b))
            ((and (< a (car lst)) (> b (car lst))) (cons (car
               lst) (fit (cdr lst) a b)))
            (t (fit (cdr lst) a b))
      )
)
```

1.6 Задание 6

Написать рекурсивную версию (с именем rec-add) вычисления суммы чисел заданного списка:

— одноуровневого смешанного,

— структурированного.

```
(defun rec-add (lst res)
      (cond ((null lst) res)
            ((listp (car lst)) (rec-add (car lst) res))
```

```
            ((numberp (car lst)) (rec-add (cdr lst) (+ res (car
               lst))))
            (t (rec-add (cdr lst) res))
      )
)
```

1.7 Задание 7

Написать рекурсивную версию с именем recnth функции nth.

```
(defun recnth (n lst)
    (cond ((null lst) Nil)
          ((< (length lst) (- n 1)) Nil)
          ((= n 0) (car lst))
          (t (recnth (- n 1) (cdr lst)))
    )
)
```

1.8 Задание 8

Написать рекурсивную функцию allodd, которая возвращает t, когда все элементы списка нечетные.

```
(defun allodd (lst)
    (cond ((null lst) t)
          ((listp (car lst)) (allodd (car lst)))
          ((oddp (car lst)) (allodd (cdr lst)))
          (t Nil)
    )
)
```

1.9 Задание 9

Написать рекурсивную функцию, которая возвращает первое нечетное число из списка, возможно, создавая некоторые вспомогательные функции.

```
(defun first-odd (lst)
    (cond ((null lst) Nil)
          ((listp (car lst)) (first-odd (car lst)))
          ((oddp (car lst)) (car lst))
          (t (first-odd (cdr lst)))
    )
)
```

1.10 Задание 10

Используя cons-дополняемую функцию с одним тестом завершения, написать функцию, которая получает как аргумент список чисел, а возвращает список квадратов этих чисел в том же порядке.

```
(defun square (lst)
    (cond ((null lst) Nil)
          (t (cons (* (car lst) (car lst)) (square (cdr lst))))
    )
)
```