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 ИУ7-66Б _____ Жаворонкова А. А.
(Группа) (Подпись, дата) (И. О. Фамилия)

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

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

2024 г.

Практическая часть

Листинг 1 – Исходный код заданий лабораторной работы

```
;; 1. Функция my-reverse, которая разворачивает верхний уровень своего списка-аргумента
(defun _my-reverse(lst res)
    (cond ((null lst) res)
          (t (_my-reverse (cdr lst) (cons (car lst) res)))))

(defun my-reverse(lst)
    (_my-reverse lst Nil))

;; 2. Функция, которая возвращает первый элемент списка-аргумента, который сам является непустым списком
(defun first-element(lst)
    (cond ((atom lst) lst)
          (t (first-element (car lst)))))

;; 3. Функция, которая выбирает из заданного списка только те числа, которые между 1 и 10
(defun is-between-1-10(x)
    (and (numberp x)
         (and (< x 10) (> x 1))))

(defun select-between-1-10(lst)
    (cond ((null lst) Nil)
          ((is-between-1-10 (car lst))
                    (cons (car lst) (select-between-1-10 (cdr lst))))
          (t (select-between-1-10 (cdr lst)))))

;; 4. Функция, которая умножает на заданное число-аргумент все числа из заданного списка-аргумента, когда:
;;       а) все элементы списка --- числа;
(defun multiply-list-1(lst x)
    (cond ((null lst) Nil)
          (t (cons (* (car lst) x)
                   (multiply-list-1 (cdr lst) x)))))

;;       б) элементы списка --- любые объекты.
(defun multiply-list-2(lst x)
```

```
34      (cond ((null lst) Nil)
35            ((numberp (car lst))
36                  (cons (* (car lst) x)
37                        (multiply-list-2 (cdr lst) x)))
38            ((listp (car lst))
39                  (cons (multiply-list-2 (car lst) x)
40                        (multiply-list-2 (cdr lst) x)))
41            (t (cons (car lst)
42                     (multiply-list-2 (cdr lst) x)))))
43
44  ;; 5. Функция 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 (доп.: упорядоченного по
       возрастанию)
45  (defun my-sort(lst)
46      (let ((tmp 0))
47           (cond ((null lst) Nil)
48                 (t (cons (and (setf tmp (apply #'min lst))
49                               (setf (car (member tmp lst)) (car
                                    lst))
50                               (setf (car lst) tmp))
51                          (my-sort (cdr lst)))))))
52
53  (defun select-between(lst a b)
54      (cond ((null lst) Nil)
55            ((or (and (< a (car lst))
56                      (> b (car lst)))
57                 (and (< b (car lst))
58                      (> a (car lst))))
59                          (my-sort (cons (car lst)
                                      (select-between (cdr lst) a b))))
60            (t (select-between (cdr lst) a b))))
61
62  ;; 6. Функция rec-add, вычисляющая сумму чисел заданного списка:
63  ;;       a) одноуровнего смешанного;
64  (defun rec-add-1(lst)
65      (cond ((null lst) 0)
66            ((numberp (car lst))
67                  (+ (car lst) (rec-add-1 (cdr lst))))
68            (t (rec-add-1 (cdr lst)))))
```

```
69
70  ;;      6) структурированного.
71  (defun rec-add-2(lst)
72      (cond ((null lst) 0)
73            ((numberp (car lst))
74                  (+ (car lst) (rec-add-2 (cdr lst))))
75            ((listp (car lst))
76                  (+ (rec-add-2 (car lst))
77                     (rec-add-2 (cdr lst))))
78            (t (rec-add-2 (cdr lst)))))
79
80  ;; 7. Функция recnth --- рекурсивная версия nth
81  (defun recnth(n lst)
82      (cond ((= n 0) (car lst))
83            (t (recnth (- n 1) (cdr lst)))))
84
85  ;; 8. Функция allodd, которая возвращает t, когда все элементы
       списка нечетные
86  (defun allodd(lst)
87      (cond ((null lst) T)
88            ((evenp (car lst)) Nil)
89            (t (allodd (cdr lst)))))
90
91  ;; 9. Функция, которая возвращает первое нечетное число из
       структурированного списка
92  (defun first-odd(lst)
93      (cond ((null lst) Nil)
94            ((and (numberp lst) (oddp lst))
95                      lst)
96            ((atom lst) Nil)
97            (t (or (first-odd (car lst))
98                   (first-odd (cdr lst))))))
99
100 ;; 10. Используя cons-дополняемую рекурсию с одним тестом
       завершения, написать функцию, которая получает как аргумент
       список чисел, а возвращает список квадратов этих чисел в том
       же порядке
101 (defun squared-list(lst)
102     (cond ((null lst) Nil)
103           (t (cons (* (car lst) (car lst))
104                    (squared-list (cdr lst))))))
```