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 № 5

по курсу «Функциональное и логическое программирование»

на тему: «Использование функционалов»

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, которая уменьшает на 10 все числа из
   списка-аргумента этой функции, проходя по верхнему уровню
   списковых ячеек (список смешанный структурированный)
(defun minus10-list(lst)
    (mapcar #'(lambda(x)
                    (if (numberp x) (- x 10) x)) lst))

;; 2. Функция, которая получает как аргумент список чисел, а
   возвращает список квадратов этих чисел в том же порядке
(defun square-list(lst)
    (mapcar #'(lambda(x) (* x x)) lst))

;; 3. Функция, которая умножает на заданное число-аргумент все
   числа из заданного списка-аргумента, когда
;;      а) все элементы списка --- числа
(defun multiply-list(lst num)
    (mapcar #'(lambda(x) (* num x)) lst))

;;      б) элементы списка --- любые объекты
(defun multiply-list-common(lst num)
    (mapcar #'(lambda(x)
                    (if (numberp x) (* x num) x)) lst))

;; 4. Функция, которая по своему списку-аргументу определяет
   является ли он палиндромом, для одноуровнего смешанного списка
(defun is-palindrom(lst)
    (every #'(lambda(x) x)
            (mapcar #'(lambda(x y) (eql x y))
                        lst (reverse lst))))

;; 5. Используя функционалы, написать предикат set-equal,
   который возвращает T, если 2 его множества-аргумента
   (одноуровневые списки) содержат одни и те же элементы,
   порядок которых не имеет значения
(defun set-equal(set1 set2)
    (every #'(lambda(x) (= x 1))
            (mapcar #'(lambda(el) (if (member el set2) 1 0))
                    set1)))
```

```
31
32 ;; 6. Функция select-between, которая из списка-аргумента,
      содержащего только числа, выбирает только те, которые
      расположены между двумя указанными числами ---
      границами-аргументами и возвращает их в виде списка (доп:
      упорядоченного по возрастанию)
33 (defun sort-list(lst)
34     (let ((temp 0))
35          (maplist #'(lambda(x)
36                 (and (setf temp (apply #'min x))
37                      (setf (car (member temp x)) (car x))
38                      (setf (car x) temp))) lst)))
39
40 (defun select-between(x y lst)
41     (sort-list (mapcan #'(lambda(el) (if (or (and (> x el) (< y
          el))
42                                      (and (< x el) (> y el)))
43                                 (cons el ())
44                                 ())) lst)))
45
46 ;; 7. Функция, вычисляющая декартово произведение двух своих
      списков-аргументов
47 (defun decart(lstX lstY)
48     (mapcan #'(lambda(x)
49                 (mapcar #'(lambda(y) (list x y)) lstY))
50                         lstX))
51
52 ;; 8. Почему так реализовано reduce, в чем причина?
53 (reduce #'+ ())      ;; --> 0
54 (reduce #'* ())      ;; --> 1
55 ;; Потому что reduce применяет первый аргумент к элементам
      второго аргумента-списка каскадным образом. Так как списки
      пустые, reduce возвращает начальное значение --- в данном
      случае нейтральный элемент по сложению и умножению
56
57 ;; 9. Функция, которая вычисляет сумму длин всех элементов
      списка, состоящего из списков (количество атомов)
58 (defun len-list-of-lists(lst)
59     (apply #'+ (mapcar #'(lambda(el)
60                              (length el)) lst)))
```