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 ИУ7-63Б _____ Лысцев Н. Д.
(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.

1 Практические задания

1.1 Задание 1

Напишите функцию, которая уменьшает на 10 все числа из списка-аргумента этой функции, проходя по верхнему уровню списковых ячеек.

Листинг 1.1 – Решение задания №1

```
(defun f1-for-elem(elem)
  (cond ((numberp elem) (- elem 10))
    (t elem)))

(defun f1(lst)
  (mapcar #'f1-for-elem lst))
```

1.2 Задание 2

Написать функцию, которая получает как аргумент список чисел, а возвращает список квадратов этих чисел в том же порядке.

Листинг 1.2 – Решение задания №2

```
(defun f2(lst)
  (mapcar #'(lambda(elem)
          (cond ((numberp elem) (* elem elem))
            (t elem))) lst))
```

1.3 Задание 3

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

a) все элементы списка - числа,

b) элементы списка - любые объекты.

Листинг 1.3 – Решение задания №3 (начало)

```
(defun f3-1(n lst)
  (mapcar #'(lambda(elem)
          (* elem n)) lst))
```

Листинг 1.4 – Решение задания №3 (конец)

```
(defun f3-2(n lst)
  (mapcar #'(lambda(elem)
          (cond ((numberp elem) (* elem n))
            (t elem))) lst))
```

1.4 Задание 4

Написать функцию, которая по своему списку-аргументу `lst` определяет является ди он палиндромом (то есть равны ли `lst` и `(reverse lst)`), для одноуровнего смешанного списка.

Листинг 1.5 – Решение задания №4

```
(defun cmp-list(lst1 lst2)
  (reduce #'(lambda(val1 val2)
          (and val1 val2)) (mapcar #'eql lst1 lst2)))

(defun is-palindrome(lst)
  (apply #'(lambda(lst1 lst2)
         (cmp-list lst1 lst2)) (list lst (reverse lst))))
```

1.5 Задание 5

Используя функционалы, написать предикат `set-equal`, который возвращает t, если два его множества-аргумента содержат одни и те же элементы, порядок которых не имеет значения.

Листинг 1.6 – Решение задания №5 (начало)

```
(defun apply-or(lst)
  (reduce #'(lambda(val1 val2)
          (or val1 val2)) lst))

(defun apply-and(lst)
  (reduce #'(lambda(val1 val2)
          (and val1 val2)) lst))

(defun -set-equal(lst1 lst2)
  (apply-and (mapcar #'(lambda(elem1)
          (apply-or (mapcar #'(lambda(elem2)
                      (eql elem1 elem2)) lst2))) lst1)))
```

Листинг 1.7 – Решение задания №5 (конец)

```
(defun set-equal(lst1 lst2)
  (let ((len1 (length lst1))
    (len2 (length lst2)))
    (cond ((not (= len1 len2)) nil)
      (t (-set-equal lst1 lst2)))))

(defun _set-equal(lst1 lst2)
  (let ((inters (intersection lst1 lst2))
    (len1 (length lst1))
    (len2 (length lst2)))
    (and (= len1 len2)
     (= len2 (length inters)))))
```

1.6 Задание 6

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

Листинг 1.8 – Решение задания №6

```
(defun select-between(n m lst)
  (remove-if #'(lambda(elem)
         (or (< elem n) (> elem m))) lst))
```

1.7 Задание 7

Написать функцию, вычисляющую декартово произведение двух своих списков-аргументов.

Листинг 1.9 – Решение задания №7

```
(defun decart(lstX lstY)
  (mapcan #'(lambda(x)
          (mapcar (lambda(y)
            (list x y)) lstY)) lstX))
```

1.8 Задание 8

Почему так реализовано `reduce`, в чем причина?

Листинг 1.10 – Решение задания №8

```
(reduce #'+ ()) -> 0
(reduce #'* ()) -> 1
```

Результаты в данном случае обусловлены тем, что в функции reduce в Common Lisp, если последовательность пуста, то возвращается начальное значение, указанное как аргумент `:initial-value`. В противном случае возвращается начальное значение первого элемента последовательности.

Поэтому, когда `reduce` применяется к пустой последовательности, начальное значение становится результатом.

Таким образом:

- `(reduce #'+ '())` возвращает 0, так как начальное значение для сложения равно 0;
- `(reduce #'* '())` возвращает 1, так как начальное значение для умножения равно 1.

1.9 Задание 9

Пусть `list-of-list` список, состоящий из списков. Написать функцию, которая вычисляет сумму длин всех элементов.

Листинг 1.11 – Решение задания №9

```
(defun sum-all-num(list-of-list)
  (reduce #'+ (mapcar #'(lambda(list)
              (length list)) list-of-list)))
```