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 №4

по курсу «Функциональное и логическое программирование»

на тему: «Использование функционалов»

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.

СОДЕРЖАНИЕ

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.

```
(defun f1 (lst)
    (cond ((null lst) Nil)
          (t (cond
                ((numberp (car lst)) (cons (- (car lst) 10) (f1
                   (cdr lst))))
                (t (cons (car lst) (f1 (cdr lst))))
          ))
    )
)

(defun f2 (lst)
    (mapcar #'(lambda (x) (cond ((numberp x) (- x 10))
                                (t x))) lst)
)
```

1.2 Задание 2

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

```
(defun f (lst)
    (mapcar #' * lst lst)
)

(defun f (lst)
    (mapcar #'(lambda (x) (* x x)) lst)
)
```

1.3 Задание 3

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

— все элементы — числа,

— элементы списка — любые объекты.

```
(defun f1 (lst n)
    (cond ((null lst) Nil)
          (t (cond
                ((numberp (car lst)) (cons (* (car lst) n) (f1
                   (cdr lst) n)))
                (t (cons (car lst) (f1 (cdr lst) n)))
          ))
    )
)

(defun f2 (lst n)
    (mapcar #'(lambda (x) (cond ((numberp x) (* x n))
                                (t x))) 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.

```
(defun f (lst)
    (equal (reverse lst) 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.

```
(defun set-equal (lst1 lst2)
    (let
        ((inter (intersection lst1 lst2)))
    (cond ((and (= (length lst1) (length lst2))
                (= (length lst1) (length inter)))
           t)
          (t nil)))
)
```

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).

```
(defun select-between (lst num1 num2)
  (cond ((null lst) Nil)
        (t (reduce #'(lambda (x y)
                                  (cond ((and (> y num1) (< y
                                     num2))
                                       (cons y x))
                                      (t x)
                                  )
        ) lst :initial-value nil))
  )
)
```

1.7 Задание 7

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

```
(defun f (lstX lstY)
    (apply #'append
        (mapcar #'(lambda (x)
                            (mapcar #'(lambda (y) (cons x (cons
                               y Nil))) lstX)) lstY))
)
```

1.8 Задание 8

Почему так реализовано `reduce`, в чем причина?

```
(print (reduce #'+ ())) ;; 0
(print (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.

```
(defun f (lst n)
    (cond ((null lst) n)
          ((listp (car lst)) (f (car lst) n))
          (t (f (cdr lst) (+ n 1)))
    )
)

(defun list-of-list(lst)
    (f lst 0)
)
```