



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 ИУ7-61Б
(Группа)

(Подпись, дата)

Савинова М. Г.
(Фамилия И. О.)

Преподаватель

(Подпись, дата)

Толпинская Н. Б.
(Фамилия И. О.)

Преподаватель

(Подпись, дата)

Строганов Ю.В.
(Фамилия И. О.)

2024 г.

СОДЕРЖАНИЕ

1	Практические задания	3
1.1	Задание 1	3
1.2	Задание 2	3
1.3	Задание 3	3
1.4	Задание 4	4
1.5	Задание 5	4
1.6	Задание 6	5
1.7	Задание 7	5
1.8	Задание 8	5
1.9	Задание 9	6

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.

```
1 (defun f1 (lst)
2   (cond ((null lst) Nil)
3         (t (cond
4              ((numberp (car lst)) (cons (- (car lst) 10) (f1
5                                              (cdr lst))))
6              (t (cons (car lst) (f1 (cdr lst))))
7            ))
8   )
9
10 (defun f2 (lst)
11   (mapcar #'(lambda (x) (cond ((numberp x) (- x 10))
12                               (t x))) lst)
13 )
```

1.2 Задание 2

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

```
1 (defun f (lst)
2   (mapcar #' * lst lst)
3 )
4
5 (defun f (lst)
6   (mapcar #'(lambda (x) (* x x)) lst)
7 )
```

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.

```

1 (defun f1 (lst n)
2   (cond ((null lst) Nil)
3         (t (cond
4              ((numberp (car lst)) (cons (* (car lst) n) (f1
5                                           (cdr lst) n)))
6              (t (cons (car lst) (f1 (cdr lst) n))))
7   ))
8 )
9
10 (defun f2 (lst n)
11   (mapcar #'(lambda (x) (cond ((numberp x) (* x n))
12                               (t x))) lst)
13 )

```

1.4 Задание 4

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

```

1 (defun f (lst)
2   (equal (reverse lst) lst)
3 )

```

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.

```

1 (defun set-equal (lst1 lst2)
2   (let
3     ((inter (intersection lst1 lst2)))
4     (cond ((and (= (length lst1) (length lst2))
5                  (= (length lst1) (length inter)))
6           t)
7           (t nil)))
8 )

```

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).

```
1 (defun select-between (lst num1 num2)
2   (cond ((null lst) Nil)
3         (t (reduce #'(lambda (x y)
4                       (cond ((and (> y num1) (< y
5                                   num2))
6                             (cons y x))
7                             (t x)
8                       )
9         ) lst :initial-value nil)))
10 )
```

1.7 Задание 7

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

```
1 (defun f (lstX lstY)
2   (apply #'append
3         (mapcar #'(lambda (x)
4                     (mapcar #'(lambda (y) (cons x (cons
5                                   y Nil)))) lstX)) lstY))
5 )
```

1.8 Задание 8

Почему так реализовано `reduce`, в чем причина?

```
1 (print (reduce #'+ ())) ;; 0
2 (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.

```
1 (defun f (lst n)
2   (cond ((null lst) n)
3         ((listp (car lst)) (f (car lst) n))
4         (t (f (cdr lst) (+ n 1))))
5   )
6 )
7
8 (defun list-of-list(lst)
9   (f lst 0)
10 )
```