



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, искусственный интеллект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №5
по курсу «Функциональное и логическое
программирование»
«Использование функционалов»

Студент группы **ИУ7-64Б**

(Подпись, дата)

Д.С. Чепиго

(И.О. Фамилия)

Преподаватели

(Подпись, дата)

Н.Б. Толпинская

(И.О. Фамилия)

(Подпись, дата)

Ю.В. Строганов

(И.О. Фамилия)

2023 г.

Содержание

1	Практические задания	3
1.1	Задание №1	3
1.2	Задание №2	3
1.3	Задание №3	4
1.4	Задание №4	4
1.5	Задание №5	5
1.6	Задание №6	5
1.7	Задание №7	6
1.8	Задание №8	6
1.9	Задание №9	7

1 Практические задания

Используя функционалы:

1.1 Задание №1

Напишите функцию, которая уменьшает на 10 все числа из списка-аргумента этой функции, проходя по верхнему уровню списковых ячеек. (* Список смешанный структурированный)

```
1 (defun all-num-minus-10 (lst)
2   (mapcar #'(lambda (x)
3     (cond
4       ((null x) NIL)
5       ((numberp x) (- x 10))
6       ((listp x) (all-num-minus-10 x))
7       (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.

```
1 (defun sqaure-numbers (lst)
2   (mapcar #'(lambda (x)
3     (* x x))
4   lst))
```

1.3 Задание №3

Напишите функцию, которая умножает на заданное число-аргумент все числа из заданного списка-аргумента, когда: все элементы списка – числа, элементы списка – любые объекты.

```
1 (defun all-numbers (lst num)
2   (mapcar #'(lambda (x)
3               (* x num))
4   lst))
5
6 (defun not-all-numbers (lst num)
7   (mapcar #'(lambda (x)
8               (cond
9                 ((null x) NIL)
10                ((numberp x) (* x num))
11                ((listp x) (not-all-numbers x num))
12                (T x)))
13   lst))
```

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 reverse1 (lst)
2   (when lst
3     (append (reverse1 (cdr lst))
4     (list (car lst)))))
5
6 (defun palyndrom-list (lst)
7   (reduce #'(lambda (x y) (and x y))
8   (mapcar #'equal lst (reverse1 lst))))
```

1.5 Задание №5

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

```
1 (defun set-size (set1 &optional (size 0))
2   (cond ((null set1) size)
3   (T (set-size (cdr set1) (+ size 1)) )))
4
5 (defun set-include (set1 set2)
6   (reduce #'(lambda (x y) (and x y))
7   (mapcar #'(lambda (x)
8     (reduce #'(lambda (x y) (or x y))
9     x))
10  (mapcar #'(lambda (elem)
11    (maplist #'(lambda (x)
12      (equalp (car x) elem))
13    set2))
14  set1))))
15
16 (defun set-equal (set1 set2)
17   (and (= (set-size set1) (set-size set2))
18   (set-include set1 set2) (set-include set2 set1)))
```

1.6 Задание №6

Напишите функцию, `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 (упорядоченного по возрастанию (+ 2 балла)).

```
1 (defun select-between (num1 num2 lst)
2   (cond ((null lst) NIL)
3   (T (reduce #'(lambda (x y)
```

```

4         (if (< num1 y num2)
5             (cons y x)
6             x))
7     lst :initial-value NIL))))

```

1.7 Задание №7

Написать функцию, вычисляющую декартово произведение двух своих списков- аргументов. (Напомним, что $A \times B$ это множество всевозможных пар (a, b) , где a принадлежит A , принадлежит B .)

```

1 (defun decart (lst1 lst2)
2   (mapcar #'(lambda (x)
3     (mapcar #'(lambda (y) (list x y))
4               lst2))
5   lst1))

```

1.8 Задание №8

Почему так реализовано `reduce`, в чем причина?

$(\text{reduce}\#' + ()) \rightarrow 0$

Поведение в данном примере обусловлено работой функции `+`. Эта функция – функционал, который при 0 количестве аргументов возвращает значение 0. Если подать на вход `reduce` функцию, которая не может обработать 0 аргументов, то вызов `reduce` с пустым списком в качестве второго аргумента вернет ошибку. При этом, если подано более одного аргумента, то `reduce` выполняет следующие действия:

- сохраняет первый элемент списка в область памяти (асс);
- для всех остальных элементов списка выполняет переданную в качестве первого аргумента функцию, подавая на вход 2 аргумента (асс и очередной элемент списка) и сохраняя результат в асс.

$(\text{reduce}\#' * ()) \rightarrow 1$

Для умножения ситуация аналогичная.

1.9 Задание №9

Пусть list-of-list список, состоящий из списков. Написать функцию, которая вычисляет сумму длин всех элементов list-of-list (количество атомов), т.е. например для аргумента ((1 2) (3 4)) -> 4.

```
1 (defun set-size (set1 &optional (size 0))  
2   (cond ((null set1) size)  
3   (T (set-size (cdr set1) (+ size 1)) )))  
4  
5 (defun len-list-of-list (lst)  
6   (cond ((null lst) 0)  
7   (T (reduce #'+ (mapcar #'set-size lst))))))
```