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»

НАПРАВЛЕНИЕ ПОДГОТОВКИ «09.03.04 Программная инженерия»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

Название: Использование функционалов

Дисциплина: Функциональное и логическое программирование

Студент	ИУ7-66Б		Т. А. Казаева
	Группа	Подпись, дата	И. О. Фамилия
Преподаватель			Н. Б. Толпинская
		Подпись, дата	И. О. Фамилия

Москва, 2022 г.

1. Практические задания

Используя функционалы:

1. *Напишите функцию, которая уменьшает на 10 все числа из списка-аргумента этой функции.*

```
(defun subtract-dec(lst)
  (mapcar #'(lambda (elem) (cond ((numberp elem)(- elem 10))
                                 ((listp elem)(subtract-dec elem))
                                 (t elem))) lst))
```

2. *Напишите функцию, которая умножает на заданное число-аргумент все числа из заданного списка-аргумента, когда:*

   a) *все элементы списка — числа,*

   b) *элементы списка – любые объекты.*

```
(defun mul-all(m lst)
  (mapcar #'(lambda (elem) (cond ((numberp elem)(* m elem))
                                 ((listp elem)(mul-all m elem))
                                 (t elem))) lst))
```

3. *Написать функцию, которая по своему списку-аргументу lst определяет является ли он палиндромом (то есть равны ли lst и (reverse lst)).*

```
(defun polyp(lst)
  (equal lst (reverse lst)))
```

4. *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-equalp (set1 set2)
(and (= (length set1) (length set2))
     (every #'(lambda (elem) (member elem set2 :test #'equal)) set1)
     (every #'(lambda (elem) (member elem set1 :test #'equal)) set2)))
```

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

```
(defun make-square(lst)
  (mapcar #'(lambda (elem) (* elem elem)) lst))
```

6. *Напишите функцию, select-between, которая из списка-аргумента, содержащего только числа, выбирает только те, которые расположены между двумя указанными границами-аргументами и возвращает их в виде списка (упорядоченного по возрастанию списка чисел (+ 2 балла)).*

```
(defun select-between (lst left right)
  (sort (reduce #'(lambda (res el) (if (and (> el left) (< el right))
                                       (cons el res)
                                           res)) lst :initial-value ()) #'<))
```

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

```
(defun combinations(lst1 lst2)
  (mapcan #'(lambda (inner)
    (mapcar #'(lambda (outer)(list outer inner)) lst1)) lst2))
```

8. *Почему так реализовано reduce, в чем причина?*

a) (**reduce** #'+ 0)-> 0

b) (**reduce** #'+ 0)-> (**reduce** #'+ ())-> 0

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

```
(defun inner-len(lists)
  (apply #'+ (mapcar #'(lambda (elem)(cond ((listp elem) (inner-len elem))
                                           (t 1))) lists)))
```