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»

Отчёт по лабораторной работе №6 по курсу

«Функциональное и логическое программирование»

Тема Использование функционалов.

Студент Сироткина П.Ю.

Группа ИУ7-66Б

Преподаватели Толпинская Н.Б., Строганов Ю.В.

Москва — 2022 г.

Практические задания

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

```
(defun dec_10(lst)
      (mapcar #'(lambda (elem)
                  (cond ((numberp elem) (- elem 10)) (T elem))) lst))
```

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

1. Все элементы списка – числа;
2. Элементы списка – любые объекты.

```
(defun mul_num(lst multiplier)
    (mapcar #'(lambda(elem) (* elem multiplier)) lst))

(defun mul(lst multiplier)
    (mapcar #'(lambda (elem)
                        (cond ((listp elem) (mul (cdr elem) multiplier))
                        ((numberp elem) (* elem multiplier))
                        (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 my_reverse(lst)
    (reduce #'(lambda (res lst) (cons lst res)) lst :initial-value ()))

(defun palindrome(lst)
    (equal lst (my_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 in_set(elem src_set)
      (reduce #'(lambda (x y) (or x y))
          (mapcar #'(lambda (x) (equal x elem)) src_set)
          :initial-value Nil))

(defun is_subset(set1 set2)
      (reduce #'(lambda (x y) (and x y))
          (mapcar #'(lambda (x) (in_set x set2)) set1) :initial-value T))

(defun set_equal(set1 set2)
      (and (is_subset set1 set2) (is_subset set2 set1)))
```

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

```
(defun make_squares(lst) (mapcar #'(lambda (elem) (* elem elem)) lst))
```

6. Написать 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_lst elem)
              (if (< left elem right) (cons elem res_lst) res_lst))
                  lst :initial-value ()) #'<))
```

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

```
(defun decart(lst_x lst_y)
      (mapcan #'(lambda(x) (mapcar #'(lambda(y) (list x y)) lst_y)) lst_x))
```

8. Почему так реализовано reduce, в чем причина?

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

Начальный результат работы функционала **reduce** - т.н. нейтральный. Для операции сумма нейтральным элементом является 0.

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

```
(defun sum_lengths(lol)
    (reduce #'(lambda(cur_sum lst) (+ cur_sum (length lst))) (cons 0 lol)))
```