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 №5

Название: Использование функционалов

Дисциплина: Функциональное и логическое программирование

Студент ИУ7-66Б _____ А.Д. Ковель
Группа Подпись, дата И. О. Фамилия

Преподаватель _____ Н. Б. Толпинская

Преподаватель _____ Ю. В. Строганов
Подпись, дата И. О. Фамилия

Москва, 2023 г.

1 Практические задания

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

```
(defun minus-ten-list (lst) (
  mapcar #'(
        lambda (elem) (
          cond
            ((numberp elem) (- elem 10))
            (T elem)
        )
      ) lst
  )
)
```

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

```
(defun square-lst (lst) (
  mapcar #'(
        lambda (elem) (
          cond
            ((numberp elem) (* elem elem))
            (T elem)
        )
      ) lst
  )
)
```

3. *Напишите функцию, которая умножает на заданное число-аргумент все числа из заданного списка-аргумента, когда а) все элементы списка — числа, б) элементы списка – любые объекты.*

```

(defun multiply-lst-numbers (lst x) (
  mapcar #'(
        lambda (elem) (
          * elem x
```

```
            )
          ) lst
      )
    )


    (defun multiply-lst (lst x) (
      mapcar #'(
            lambda (elem) (
              cond
                ((numberp elem) (* elem x))
                (T elem)
            )
          ) lst
      )
    )
```

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

```
    (defun polyp (lst) (
      cond ((eql
        (find-if-not #'oddp

          (mapcar #'(
            lambda (elem revelem) (
              cond
                ((eql elem revelem) 1)
                (T 0)
            )
            ) lst (reverse lst)
          )
        )
        Nil) '(T))
        (T Nil)
    )
    )
```

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.

```
(defun set-equalp (set1 set2)
(
  and (eql (length set1) (length set2))
       (every #'(lambda (elem)
                  (member elem set2 :test #'equal)
                ) set1)
       (every #'(lambda (elem)
                  (member elem set1 :test #'equal)
                ) set2)
)
)
```

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)).*

```
(defun select-between (lst l r)
(
  sort (reduce #'(
          lambda (res el) (
            cond ((and (> el l) (< el r)) (cons el res))
                 (T res)

          )
          )
  lst :initial-value ()) #'<
)
)
```

7. *Написать функцию, вычисляющую декартово произведение двух своих спискова́ргументов. (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? (reduce #'+ ()) -> 0 (reduce #'* ()) -> 1*

```
1    (reduce #'+ ()) -> 0
2    (reduce #'* ()) -> 1
```

начальное занчение функции + — 0

начальное значение функции * — 1

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 list-of-list (lst)
2      (apply #'+ (
3             mapcar #'(
4               lambda (elem) (
5                 cond ((listp elem) (list-of-list elem))
6                   (t 1)
7               )
8             )
9             lst
10          )
11     )
12   )
```