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 №5 по курсу «Функциональное и логическое программирование»

Тема Использование функционалов

Студент Егорова П.А.

Группа ИУ7-64Б

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

Москва — 2023 г.

1. Практические задания

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

```
(defun substract(lst)
    (mapcar #'(lambda (elem)
                      (if (numberp elem)
                          (- elem 10)
                          elem
                      )
              )
              lst
    )
)
```

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

```
(defun square(lst)
    (mapcar #'(lambda (x)
                      (* x x)
              )
              lst
    )
)
```

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

1.3.1. Все элементы списка — числа

```
(defun mult1 (array koef)
    (mapcar #'(lambda (elem) (* elem koef))
            array
    )
)
```

1.3.2. Элементы списка – любые объекты.

```
(defun mult2 (array koef)
    (mapcar #'(lambda (elem)
                        (if (numberp elem)
                            (* elem koef)
                            elem
                        )
                )
            array
    )
)
```

1.4. 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.

```
(defun is_palindrom (lst)
    (equal lst (reverse lst))
)
```

1.5. 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-equal(lst1 lst2)
    (and (subsetp lst1 lst2)
         (subsetp lst2 lst1)
    )
)
```

1.6. 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 (a b lst)
    (sort
        (mapcan #'(lambda (x)
                          (and (>= x a) (<= x b) (list x))
                  )
                lst
        )
        #'<
    )
)
```

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

```
(defun decart(lstX lstY)
    (mapcar #'(lambda (x)
                      (mapcar #'(lambda (y) (list x y))
                              lstY
                      )
              )
            lstX
    )
)
```

1.8. Почему так реализовано reduce, в чем причина?

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

Синтаксис reduce:

```
(reduce function proseq
    &key key from-end start end initial-value)
```

Если proseq пуста и предоставлено initial-value, оно возвращается, но если initial-value не предоставлено, функция вызывается без аргументов. То есть

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

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.

```
(defun len(lst)
    (reduce #'(lambda (current elem)
                      (+ current (length elem))
              )
              (cons 0 lst)
    )
)
```