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** _____ **Д.С. Чепиго**

(Подпись, дата) (И.О. Фамилия)

Преподаватели _____ **Н.Б. Толпинская**

(Подпись, дата) (И.О. Фамилия)

_____ **Ю.В. Строганов**

(Подпись, дата) (И.О. Фамилия)

***2023** г.*

Содержание

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)

```
(defun all-num-minus-10 (lst)
    (mapcar #'(lambda (x)
        (cond
            ((not x) NIL)
            ((numberp x) (- x 10))
            ((listp x) (all-num-minus-10 x))
            (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.

```
(defun sqaure-numbers (lst)
    (mapcar #'(lambda (x)
                (* x x))
                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.

```
(defun all-numbers (lst num)
    (mapcar #'(lambda (x)
                (* x num))
                lst))

(defun not-all-numbers (lst num)
    (mapcar #'(lambda (x)
                (cond
                    ((not x) NIL)
                    ((numberp x) (* x num))
                    ((listp x) (not-all-numbers x num))
                    (T x)))
                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 одноуровнего смешанного списка.

```
(defun reverse1 (lst)
    (when lst
        (append (reverse1 (cdr lst))
                (list (car lst)))))

(defun palyndrom-list (lst)
    (reduce #'(lambda (x y) (and x y))
        (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.

```
(defun set-size (set1 &optional (size 0))
    (cond ((not set1) size)
    (T (set-size (cdr set1) (+ size 1)) )))

(defun set-include (set1 set2)
    (reduce #'(lambda (x y) (and x y))
        (mapcar #'(lambda (x)
                    (reduce #'(lambda (x y) (or x y))
                    x))
                (mapcar #'(lambda (elem)
                        (maplist #'(lambda (x)
                            (equalp (car x) elem))
                            set2))
                        set1))))

(defun set-equal (set1 set2)
     (and (= (set-size set1) (set-size set2))
     (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)).

```
(defun select-between (num1 num2 lst)
     (cond ((not lst) NIL)
        (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 x 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?

$(reduce\#' + ()) -> 0$

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 (acc);

— для всех остальных элементов списка выполняет переданную в качестве первого аргумента функцию, подавая на вход 2 аргумента (acc и очередной элемент списка) и сохраняя результат в acc.

$(reduce\#' * ()) -> 1$

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.

```
(defun set-size (set1 &optional (size 0))
    (cond ((not set1) size)
    (T (set-size (cdr set1) (+ size 1)) )))

(defun len-list-of-list (lst)
    (cond ((not lst) 0)
          (T (reduce #'+ (mapcar #'set-size lst)))))
```