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»

ОТЧЕТ

по Лабораторной работе №6

по курсу «Функциональное и логическое программирование»

на тему: «Использование функционалов»

Студент ИУ7-63Б _____ Недолужко Д. В.
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ Толпинская Н. Б.
(Подпись, дата) (И. О. Фамилия)

2022 г.

1 Практическая часть

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

Листинг 1.1 – Функция, проверяющая, является ли список палиндромом

```
1 (defun minus-10 (lst)
2   (mapcar (lambda (x) (- x 10)) lst))
```

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

1.2.1 а) все элементы списка – числа

Листинг 1.2 – Функция, умножающая каждый элемент списка на число

```
1 (defun mult-all-numbers (mult lst)
2   (mapcar #'(lambda (el) (* el mult)) lst))
```

1.2.2 б) элементы списка – любые объекты

Листинг 1.3 – Функция, умножающая каждое число из списка на число

```
1 (defun compl-mult-all-numbers (mult lst)
2   (mapcar
3     #'(lambda (el)
4         (cond ((listp el) (compl-mult-all-numbers mult el))
5               ((* el mult))))
6     lst))
```

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

Листинг 1.4 – Функция, возводящая все элементы списка в квадрат

```
1 (defun palindromp (lst)
2   (every #'= lst (reverse lst)))
```

1.4 Написать предикат set-equal, который возвращает t, если два его множества аргумента содержат одни и те же элементы, порядок которых не имеет значения.

Листинг 1.5 – Функция, возводящая все элементы списка в квадрат

```
1 (defun my-subsetp (s1 s2)
2   (every
3     #'(lambda (a) (some #'(lambda (b) (= a b)) s2))
4     s1))
5
6 (defun set-equal (set1 set2)
7   (cond
8     ((/= (length set1) (length set2)) nil)
9     ((my-subsetp set1 set2) (my-subsetp set2 set1))))
```

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

Листинг 1.6 – Функция, возводящая все элементы списка в квадрат

```
1 (defun sqrlist (lst)
2   (mapcar #'* lst lst))
```

1.6 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

Листинг 1.7 – Функция, выбирающая из списка-аргумента числа, расположенные между двумя указанными границами-аргументами

```
(defun select-between (from to lst)
  (mapcan #'(lambda (el)
              ((< from el to) (list el)))
          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, a b принадлежит B)

Листинг 1.8 – Функция, вычисляющая декартово произведение элементов списков аргументов

```
(defun decart (lstA lstB)
    (mapcan #'(lambda (a)
                (mapcar #'(lambda (b)
                            (list x y)) lstB))
                                        lstA))
```

1.8 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 (например, математическая функция `cons`), то вызов `reduce` с пустым списком в качестве второго аргумента вернет ошибку (`invalid number of arguments: 0`). При этом, если подано более одного аргумента, то `reduce` выполняет следующие действия:

1. сохраняет первый элемент списка в область памяти (для определенности назовем ее `acc`);

2. 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`.

Пример упрощенной реализации `reduce` (в данной реализации опущены проверки аргументов):

Листинг 1.9 – Пример реализации reduce

```
1 (defun my-reduce-internal (func lst acc)
2   (cond ((null lst) acc)
3         ((my-reduce-internal func (cdr lst) (funcall func acc (
            car lst))))))
4
5 (defun my-reduce (func lst)
6   (cond ((null lst) (funcall func))
7         ((my-reduce-internal func (cdr lst) (car lst)))))
```

1.9 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

Листинг 1.10 – Пример реализации reduce

```
1 (defun sum-lens (list-of-lists)
2   (reduce #'(lambda (acc lst) (+ acc (length lst)))
3           (cons 0 list-of-lists)))
```