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

Название: Использование функционалов

Дисциплина: Функциональное и логическое программирование

Студент	ИУ7-64Б		Л.Е.Тартыков
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Н.Б.Толпинская
		(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Ю.В.Строганов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2022

1 Практические задания

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

Листинг 1.1 – Задание 1

```
(defun subtract_by_ten (lst)
    (mapcar #'(lambda (x) (- x 10)) lst)
)
```

1.2 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.

Листинг 1.2 – Задание 2

```
(defun multiply_by_number_1 (lst number)
    (mapcar #'(lambda (x) (* x number)) lst)
)

(defun multiply_by_number_2 (lst number)
    (mapcar #'(lambda (x) (if (numberp x) (* x number)
                              (if (listp x)
                                  (multiply_by_number_1 x number)
                                  x
                              )
                          )
              ) lst)
)
```

1.3 Написать функцию, которая по своему списку-аргументу lst определяет является ли он палиндромом.

Листинг 1.3 – Задание 3

```
(defun is_palindrom (lst)
    (let ((reverse_lst (reverse lst))
          (result nil)
         )
         (setq result (mapcar #'(lambda (x y) (equal x y)) lst reverse_lst))
         (every #'(lambda (x) (not (null x))) result)
    )
)
```

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 1.4 – Задание 4

```
(defun set_equal (lst1 lst2)
    (let ((result nil))
         (cond (( /= (length lst1) (length lst2)) nil)
               (T (setq result
                  (maplist #'(lambda (x) (if (member (car x) lst2) T nil))
                     lst1)
                  )
               )
          )
          (cond ((not (null result)) (every #'(lambda (x) (eql x T)) result)))
    )
)
```

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

Листинг 1.5 – Задание 5

```
(defun square_numbers (lst)
    (mapcar #'(lambda (x) (* x x)) 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 (упорядоченного по возрастанию списка чисел).

Листинг 1.6 – Задание 6

```
(defun select_between (lst board_left board_right)
    (mapcan #'(lambda (x)
                (and (>= x board_left) (<= x board_right) (list x)))
                lst)
)

(defun bubble (lst)
    (cond ((atom (cdr lst)) lst)
          ((> (car lst) (cadr lst))
          (cons (cadr lst) (bubble (cons (car lst) (cddr lst)) ))
          )
          (T lst)
    )
)

(defun bubble_sort_asc (lst)
    (cond ((atom (cdr lst)) lst)
           (T (bubble (cons (car lst) (bubble_sort_asc (cdr lst)))))
    )
)
```

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

Листинг 1.7 – Задание 7

```
(defun cartesian_product (lst1 lst2)
    (mapcan #'(lambda (x) (mapcar #'(lambda (y) (list x y)) lst2)) lst1)
)
```

1.8 Почему так реализовано reduce, в чем причина?

Листинг 1.8 – Задание 8

```
    (reduce #'+ 0)    ; 0
    (reduce #'+ ())   ; 0
```

Функционал reduce имеет вызов по формуле (1.1):

$$(reduce\ \#'fun\ lst) \tag{1.1}$$

Reduce сначала применяет функцию fun к первым двум элементам списка; затем fun применяется к полученному результату и следующему элементу. Пример приведен на листинге 1.8.

```
    (reduce #'- '(1 2 3 4)) ; (- (- (- 1 2) 3) 4) =>  -8
```

Данный функционал может содержать начальное значение (*initial-value*) для некоторых операторов. Возможны два случая.

1. Список lst содержит ровно один элемент и начальное значение не задано; тогда этот элемент возвращается, а функция fun не вызывается.

2. Если список lst пуст, и задано начальное значение, то возвращается начальное значение, а функция не вызывается.

Пример данных ситуаций приведен на листинге 1.8.

```
    (reduce #'+ '())    ; 0 (1)
    (reduce #'+ '(3))   ; 3 (2)
    (reduce #'* '(2))   ; 2 (3)
```

Пример (1) удовлетворяет случаю (2); в итоге возвращается начальное значение оператора «+» 0. Случаю (1) удовлетворяет пример (2) и (3).

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

Листинг 1.9 – Задание 9

```
1 (defun list_of_list (lst)
2     (reduce #'(lambda (count x) (+ count (length x))) (cons 0 lst))
3 )
```