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 №7

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 Написать хвостовую рекурсивную функцию my-reverse, которая развернет верхний уровень своего списка-аргумента lst

```
1 (defun move-to (lst result)
2   (cond ((null lst)(res))
3         (T(move-to (cdr lst)(cons (car lst) res)))))
4
5 (defun my-reverse (lst)
6   (move-to lst ())
```

1.2 Написать функцию, которая возвращает первый элемент списка-аргумента, который сам является непустым списком.

```
1 (defun get-notnull(lst)
2   (cond ((null lst)Nil)
3         ((and (listp lst)(> (length lst) 0))(car lst))
4         (T(get-notnull lst))))
```

1.3 Написать функцию, которая выбирает из заданного списка только те числа, которые больше 1 и меньше 10

```
1 (defun move-to (lst res)
2   (cond ((null lst)(res))
3         ((and (numberp (car lst))(> 0 (car lst) 10))
4          (move-to (cdr lst)(cons (car lst) res)))
5         (T(move-to (cdr lst)res))))
6
7 (defun select-gt-0-lt-10 (lst)
8   (move-to lst ())
```

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

1.4.1 все элементы списка — числа

```
(defun mult-2 (lst)
  (cond ((null lst)Nil)
        (T(cons (* 2 (car lst))
                (mult-2 (cdr lst))))))
```

1.4.2 элементы списка – любые объекты

```
(defun mult-2-any (lst)
  (cond ((null lst)Nil)
        ((numberp lst)(cons (* 2 (car lst))
                            (mult-2 (cdr lst))))
        ((listp lst)(cons (mult-2-any (car lst))
                          (mult-2 (cdr lst))))))
        (T (cons (car lst)
                 (mult-2 (cdr lst))))))
```

1.5 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

```
(defun select-between (from to lst)
  (cond ((null lst)Nil)
        ((< from (car lst) to)(cons (car lst)
                                    (select-between from to (cdr
                                       lst))))
        (T(select-between from to (cdr lst)))))
```

1.6 Написать рекурсивную версию (с именем rec-add) вычисления суммы чисел заданного списка:

1.6.1 одноуровнего смешанного

```
1 (defun rec-add-internal (lst sum)
2   (cond ((null lst)sum)
3         (T(rec-add-internal (cdr lst) (+ (car lst) sum)))))
4
5 (defun rec-add (lst)
6   (rec-add-internal lst s0))
```

1.6.2 структурированного

```
1 (defun rec-add-internal (lst sum)
2   (cond ((null lst)sum)
3         ((listp lst)(rec-add-internal (car lst) (+ (
            rec-add-internal (car lst) 0) sum)))
4         (T(rec-add-internal (cdr lst) (+ (car lst) sum)))))
5
6 (defun rec-add (lst)
7   (rec-add-internal lst s0))
```

1.7 Написать рекурсивную версию с именем recnth функции nth

```
1 (defun recnth (i lst)
2   (cond ((< i 0)Nil)
3         ((= i 0)(car lst))
4         (T(recnth (- i 1)(cdr lst)))))
```

1.8 Написать рекурсивную функцию allodd, которая возвращает t когда все элементы списка нечетные.

```
(defun allodd (lst)
  (cond ((null lst)T)
        ((evenp (car lst))Nil)
        (T(allodd (cdr lst)))))
```

1.9 Используя cons-дополняемую рекурсию с одним тестом завершения, написать функцию которая получает как аргумент список чисел, а возвращает список квадратов этих чисел в том же порядке.

```
(defun mult-2 (lst)
  (cond ((null lst)Nil)
        (T(cons (* (car lst) (car lst))
                (mult-2 (cdr lst))))))
```