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: «Рекурсивные функции»

Студент ИУ7-63Б _____ Миронов Г. А.
(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

Листинг 1.1 – Функция, разворачивающая первый уровень списка

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

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

Листинг 1.2 – Функция, возвращающая первый элемент списка, являющийся списком

```
1 (defun get-first-sublist (lst)
2   (cond
3     ((null lst) nil)
4     ((cond ((listp (car lst)) (car lst))))
5     ((get-first-sublist (cdr lst)))))
```

1.3 Написать функцию, которая выбирает из заданного списка только те числа, которые больше 1 и меньше 10. (Вариант: между двумя заданными границами)

Листинг 1.3 – Функция, выбирающая из списка только те числа между двумя заданными границами

```
(defun all-between-internal (lst a b result)
  (cond
    ((null lst) result)
    ((all-between-internal
        (cdr lst)
        a
        b
        (cond
          ((cond
            ((>= (car lst) a) (<= (car lst) b)))
             (cons (car lst) result))
          (result))))))

(defun all-between (lst a b)
  (all-between-internal lst a b nil))
```

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

Листинг 1.4 – Функция, умножающая на заданное число все числа из списка чисел

```
(defun mulall-internal (lst num res)
  (cond ((null lst) res)
        ((mulall-internal
            (cdr lst)
            num
            (cons (* num (car lst)) res)))))

(defun mulall (lst num)
  (mulall-internal lst num ()))
```

1.4.2 элементы списка – любые объекты

Листинг 1.5 – Функция, умножающая на заданное число все числа из списка

```
(defun mulall-internal (lst num res)
  (cond
    ((null lst) res)
    ((symbolp (car lst))
      (mulall-internal (cdr lst) num (cons (car lst) res)))
    ((numberp (car lst))
      (mulall-internal
        (cdr lst)
        num
        (cons (* num (car lst)) res)))
    ((consp (car lst))
      (mulall-internal
        (cdr lst)
        num
        (cons
          (mulall-internal (car lst) num ())
          res)))))

(defun mulall (lst num)
  (mulall-internal lst num ()))
```

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

Листинг 1.6 – Функция, вычисляющая сумму чисел заданного одноуровневого смешанного списка

```
(defun check-border (x a b)
  (and (>= x a) (<= x b)) )

(defun select-between (lst a b)
  (cond
    ((null lst) ())
    ((check-border (car lst) a b)
      (cons (car lst) (select-between (cdr lst) a b)))
    ((select-between (cdr lst) a b))))
```

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

1.6.1 одноуровневого смешанного

Листинг 1.7 – Функция, вычисляющая сумму чисел заданного одноуровневого смешанного списка

```
(defun rec-add-internal (lst sum)
  (cond
    ((null lst) sum)
    ((rec-add-internal
      (cdr lst)
      (cond
        ((numberp (car lst)) (+ sum (car lst)))
        (sum))))))

(defun rec-add (lst)
    (rec-add-internal lst 0))
```

1.6.2 структурированного

Листинг 1.8 – Функция, вычисляющая сумму чисел заданного структурированного списка

```
(defun rec-add-internal (lst sum)
  (cond
    ((null lst) sum)
    ((numberp (car lst))
      (rec-add-internal (cdr lst) (+ sum (car lst))))
    ((null (car lst))
      (rec-add-internal (cdr lst) sum))
    ((rec-add-internal
      (cons (caar lst) (cons (cdar lst) (cdr lst)))
      sum))))

(defun rec-add (lst)
  (rec-add-internal lst 0))
```

1.7 Написать рекурсивную версию с именем recnth функции nth

Листинг 1.9 – Рекурсивная реализация функции nth

```
(defun rec-nth (n lst)
  (cond
    ((null lst) nil)
    ((= n 0) (car lst))
    ((rec-nth (- n 1) (cdr lst)))))
```

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

Листинг 1.10 – Функция, проверяющая, являются ли все элементы списка нечетными

```
(defun alloddr (lst)
  (cond
    ((null lst) T)
    ((oddp (car lst)) (alloddr (cdr lst)))))
```

1.9 Написать рекурсивную функцию, которая возвращает первое нечетное число из списка (структурированного), возможно создавая некоторые вспомогательные функции

Листинг 1.11 – Функция, возвращающая первый нечетный элемент структурированного списка

```
(defun first-odd (lst)
  (cond
    ((null lst) nil)
    ((numberp (car lst))
      (cond
        ((oddp (car lst)) (car lst))
        ((first-odd (cdr lst)))))
    ((consp (car lst))
      (first-odd (cons (caar lst) (cons (cdar lst) (cdr lst)))))
    ((first-odd (cdr lst)))))
```

1.10 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

Листинг 1.12 – Функция, возводящая все элементы списка чисел в квадрат

```
1 (defun square-lst (lst)
2   (cond
3     ((null lst) Nil)
4     ((cons
5       (* (car lst) (car lst))
6       (square-lst (cdr lst))))))
```