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 №7

Название «Рекурсивные функции»

Дисциплина «Функциональное и логическое программирование»

Студент ИУ7-65Б _____ Клименко А.К.
(подпись, дата) (Фамилия И.О.)

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

Москва, 2022

Содержание

Введение

Цель работы – приобрести навыки органинизации рекурсии в Lisp.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- изучить способы организации следующих видов рекурсии:
  - хвостовой,
  - дополняемой,
  - множественной,
  - взаимной;
- изучить способы организации рекурсии более высокого порядка в Lisp.

1 Практические задания

1.1 Задание 1

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

```
(defun my-reverse (lst)
    (funcall #'(lambda (fn) (funcall fn fn lst nil))
        (lambda (self lst res)
            (cond ((null lst) res)
                (t (funcall self self (cdr lst) (cons (car lst) res)))))))
```

1.2 Задание 2

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

```
(defun get-first-list (lst)
    (cond ((null lst) nil)
        ((and (listp (car lst)) (not (null (car lst))))
            (car lst))
        (t (get-first-list (cdr lst)))))
```

1.3 Задание 3

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

```
(defun num-between (num low high)
    (and (< low num) (< num high)))

(defun select-between (lst low high)
    (cond
        ((null lst) nil)
        ((and (< low (car lst)) (< (car lst) high))
            (cons (car lst) (select-between (cdr lst) low high)))
        (t (select-between (cdr lst) low high))))
```

1.4 Задание 4

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

a) все элементы списка – числа,

b) элементы списка – любые объекты.

```
(defun mult-a (num lst)
    (cond
        ((null lst) nil)
        (t (cons ( * num (car lst)) (mult-a num (cdr lst))))))

(defun mult-b (num lst)
    (cond
        ((null lst) nil)
        ((numberp (car lst))
            (cons ( * num (car lst)) (mult-b num (cdr lst))))
        (t (cons (car lst) (mult-b num (cdr lst))))))
```

1.5 Задание 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 (+2 балла)).

```
(defun num-between (num low high)
    (and (< low num) (< num high)))

(defun select-between (lst low high)
    (cond
        ((null lst) nil)
        ((and (< low (car lst)) (< (car lst) high))
            (cons (car lst) (select-between (cdr lst) low high)))
        (t (select-between (cdr lst) low high))))
```

1.6 Задание 6

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

a) одноуровнего смешанного,

b) структурированного.

```
(defun __rec-add-a (lst sum)
    (cond
        ((null lst) sum)
        ((numberp (car lst))
            (__rec-add-a (cdr lst) (+ sum (car lst))))
        (t (__rec-add-a (cdr lst) sum))))

(defun rec-add-a (lst)
    (__rec-add-a lst 0))

(defun __rec-add-b (lst sum)
    (cond
        ((null lst) sum)
        ((numberp (car lst))
            (__rec-add-b (cdr lst) (+ sum (car lst))))
        ((listp (car lst))
            (__rec-add-b (cdr lst) (+ sum (rec-add-b (car lst)))))
        (t (__rec-add-b (cdr lst) sum))))

(defun rec-add-b (lst)
    (__rec-add-b lst 0))
```

1.7 Задание 7

Написать рекурсивную версию с именем `recnth` функции `nth`.

```
(defun recnth (index lst)
    (cond
        ((null lst) nil)
        ((eql index 0) (car lst))
        (t (recnth (- index 1) (cdr lst)))))
```

1.8 Задание 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 Задание 9

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

```
(defun get-first-odd (lst)
    (cond
        ((null lst) nil)
        ((and (numberp (car lst)) (oddp (car lst)))
            (car lst))
        ((listp (car lst))
            (or (get-first-odd (car lst))
                (get-first-odd (cdr lst))))
        (t (get-first-odd (cdr lst)))))
```

1.10 Задание 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.

```
(defun squares (lst)
    (if (null lst) nil
        (cons ( * (car lst) (car lst))
            (squares (cdr lst)))))
```

Заключение

В ходе работы были изучены способы организации хвостовой, дополняемой, множественной и взаимной рекурсии. Также были изучены способы организации рекурсии более высокого порядка в языке Lisp.

В результате были преобретены навыки организации и работы с рекурсией в языке Lisp.