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

Лабораторная работа №10

По предмету: «Функциональное и логическое программирование»

Студент:
Коротков Андрей Владимирович
Группа: ИУ7-65Б

Преподаватели:
Толпинская Наталья Борисовна
Строганов Юрий Владимирович

Москва, 2020 г.

Задание №1. Написать функцию, которая вычисляет сумму длин всех элементов list-of-list.

```
(defun list-of-list (lst l)
 (cond ((null lst) l)
       ((atom (car lst)) (list-of-list (cdr lst) (+ l 1)))
       ((listp (car lst)) (list-of-list (cdr lst) (+ l (list-of-list (car lst) 0))))))
```

Задание №2. Написать рекурсивную версию (с именем rec-add) вычисления суммы чисел заданного списка.

```
(defun rec-add (lst rst)
 (cond ((null lst) rst)
       ((atom (car lst)) (rec-add (cdr lst) (+ rst (car lst))))
       ((listp (car lst)) (rec-add (cdr lst) (+ rst (rec-add (car lst) 0))))))

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

Задание №3. Написать рекурсивную версию с именем rec-nth функции nth.

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

Задание №4. Написать рекурсивную функцию alloddr, которая возвращает t, когда все элементы списка нечетные.

```
(defun allodr (lst rst)
  (cond ((null lst) rst)
       ((atom (car lst)) (allodr (cdr lst) (and rst (oddp (car lst)))))
       ((listp (car lst)) (allodr (cdr lst) (and rst (allodr (car lst) T))))))
```

Задание №5. Написать рекурсивную функцию, относящуюся к хвостовой рекурсии с одним тестом завершения, которая возвращает последний элемент списка-аргумента.

```
(defun my_last (curr)
  (if (null (cdr curr))
    (car curr)
    (my_last (cdr curr))
  )
)
```

Задание №6. Написать рекурсивную функцию, относящуюся к дополняемой рекурсии с одним тестом завершения, которая вычисляет сумму всех чисел от 0 до n-ого аргумента функции.
Вариант:
1) от n-аргумента функции до последнего
2) от n-аргумента до m-аргумента с шагом d

```
(defun sum-elem(lst n)
  (cond   ((null lst) 0)
          ((= n 0) 0)
          (T (+ (sum-elem (cdr lst) (- n 1)) (car lst)))
    ))

(defun sum-elem-from-n(lst n)
  (cond   ((null lst) 0)
          ((= n 0) (+ (sum-elem-from-n (cdr lst) 0) (car lst)))
          (T (sum-elem-from-n (cdr lst) (- n 1)))
    ))

(defun sum-elem-step(lst n m step)
  (cond   ((or (null lst) (<= m 0)) 0)
          ((and (<= n 0) (= (mod n step) 0) (> m 0)) (+ (sum-elem-step (cdr lst) (- n 1) (-
m 1) step) (car lst)))
          ((and (< n 0) (> m 0)) (+ (sum-elem-step (cdr lst) (- n 1) (- m 1) step) 0))
          (T (sum-elem-step (cdr lst) (- n 1) (- m 1) step))
    ))
```

Задание №7. Написать рекурсивную функцию, которая возвращает последнее нечетное число из числового списка, возможно создавая некоторые вспомогательные функции.

```
(defun get_last_odd_inner (curr value)
  (cond ((eq curr nil) value)
        ((oddp (car curr)) (get_last_odd_inner (cdr curr) (car curr)))
        (T (get_last_odd_inner (cdr curr) value))
  )
)
(defun get_last_odd (lst)
  (get_last_odd_inner lst nil)
)
```

Задание №8. 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 sqr_lst(lst)
  (if (null lst)
        nil
        (cons (* (car lst) (car lst)) (sqr_lst (cdr lst)))
  ))
```

Задание №9. Написать функцию, которая из заданного списка выбирает все нечетные числа.

```
(defun select_odd(lst)
  (cond ((null lst) nil)
     ((oddp (car lst)) (cons (car lst) (select_odd (cdr lst))))
     (T (select_odd (cdr lst)))))
```

Вычисляет сумму всех нечетных чисел из заданного списка:

```
(defun sum_odd(lst)
  (reduce #'+ (select_odd lst)))
```

Ответы на теоретические вопросы

- Способы организации повторных вычислений в Lisp?
- Есть 2 способа организации повторных вычислений:

1) Рекурсия
2) Функционал

- Различные способы использования функционалов?
- Существует две группы функционалов: применяющие и отображающие.

- Что такое рекурсия? Способы организации рекурсивных функций?
- Рекурсия — это ссылка на определяемый объект во время его определения. Т.к. в Lisp используются рекурсивно определенные структуры, то рекурсия — это естественный принцип обработки таких структур. Существуют типы рекурсивных функций: хвостовая, дополняемая, множественная, взаимная рекурсия и рекурсия более высокого порядка.

- Способы повышения эффективности реализации рекурсии.
- В целях повышения эффективности рекурсивных функций рекомендуется формировать результат не на выходе из рекурсии, а на входе в рекурсию, все

действия выполняя до ухода на следующий шаг рекурсии. Это и есть хвостовая рекурсия.