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 к лабораторной работе №7

по курсу «Функциональное и логическое программирование»

по теме «Рекурсивные функции»

Студент: Костев Д.И.

Группа: ИУ7-61Б

Преподаватель: Толпинская Н.Б.

2022 г.

Практические задания

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

Решение:

```
(defun rec_reverse (lst res)
(cond
        ((null lst) res)
        (T (rec_reverse (cdr lst) (cons (car lst) res)))))
(defun my_reverse (lst) (rec_reverse lst ()))
```

Результат работы:

(my_reverse '(1 2 3 4)) → (4 3 2 1)

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

Решение:

```
(defun return_el_list (lst)
(cond
        ((null lst))
        ((and (listp (car lst)) (not (null (car lst)))) (car lst))
        (T (return_el_list (cdr lst)))))
```

Результат работы:

(return_el_list '(1 () abc (1 2) 3)) → (1 2)

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

Решение:

```
(defun check_borders (el) (and (> el 1) (< el 10)))

(defun select_rec (lst res)

(cond
        ((null lst) res)
        ((check_borders (car lst)) (select_rec (cdr lst) (cons (car lst) res)))
        (T (select_rec (cdr lst) res))))

(defun select_between (lst) (select_rec lst ()))
```

Результат работы:

(select_between '(1 2 13 4 15 6)) → (6 4 2)

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

а) все элементы списка – числа,

б) элементы списка – любые объекты

Решение:

а)
```
(defun mult_num_rec (lst num)
   (cond
        ((null lst))
        (T (setf (car lst) (* num (car lst))) (mult_num_rec (cdr lst) num))))
        (defun mult_only_numbers (lst num) (mult_num_rec lst num) lst)
```

Результат работы:
(mult_only_numbers '(1 2 3 4) 2) → (2 4 6 8)

б)
```
(defun mult_var_rec (lst num)
   (cond
        ((null lst))
        ((numberp (car lst)) (setf (car lst) (* num (car lst)))
        (mult_var_rec (cdr lst) num)) (T (mult_var_rec (cdr lst) num))))

(defun mult_various (lst num) (mult_var_rec lst num) lst)
```

Результат работы:
(mult_various '(1 abc (2 3) 4 5 d) 4) → (4 ABC (2 3) 16 20 D)

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)).

Решение:

```
(defun check_borders (el a b) (and (> el a) (< el b)))

(defun select_rec (lst a b res)
(cond
        ((null lst) res)
        ((check_borders (car lst) a b) (select_rec (cdr lst) a b (cons (car lst) res)))
        (T (select_rec (cdr lst) a b res))))

(defun select_between (lst a b) (select_rec lst a b ()))
```

Результат работы:
(select_between '(1 2 3 4 5 6) 2 5) → (4 3)

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

а) одноуровнего смешанного,

б) структурированного.

Решение:

```
a) (defun add_1 (lst sum)
   (cond
        ((null lst) sum)
        ((numberp (car lst)) (add_1 (cdr lst) (+ sum (car lst))))
        (T (add_1 (cdr lst) sum))))

   (defun rec_add_1 (lst) (add_1 lst 0))
```

Результат работы:
(rec_add_1 '(1 2 3 4 5)) → 15
(rec_add_1 '(1 a 3 bc 5 cde)) → 9

б) (defun add_2 (lst sum)
(cond
((null lst) sum)
((numberp (car lst)) (add_2 (cdr lst) (+ sum (car lst))))
((listp (car lst)) (add_2 (cdr lst)(+ sum (add_2 (car lst) 0))))
(T (add_2 (cdr lst)sum))))

(defun rec_add_2 (lst) (add_2 lst 0))

Результат работы:
(rec_add_2 '(1 2 (3 4) (5) () 6 7)) → 28

7. Написать рекурсивную версию с именем recnth функции nth.

Решение:

```
(defun get_el_by_pos (n lst ind)
(cond
        ((null lst))
        ((= ind n) (car lst))
(T (get_el_by_pos n (cdr lst) (+ ind 1)))))

(defun recnth (n lst) (get_el_by_pos n lst 0))
```

Результат работы:
(recnth 3 '(1 2 3 4 5 6)) → 4

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

Решение:

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

Результат работы:
(all_odd '(1 3 5 7)) → T
(all_odd '(1 2 3 4 5 6 7)) → NIL

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

Решение:

```
(defun first_odd (lst)
(cond
        ((null lst))
        ((and (numberp (car lst)) (oddp (car lst))) (car lst))
        ((listp (car lst)) (or (first_odd (car lst)) (first_odd (cdr lst))))
        (T (first_odd (cdr lst)))))
```

Результат работы:
(first_odd '(2 4 6 7 3 2)) → 7
(first_odd '(2 4 (2 4) (6 5))) → 5

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.

Решение:

```
(defun square_rec (lst res)
(cond
        ((null lst) res)
        (T (square_rec (cdr lst) (append res (cons (* (car lst) (car lst)) NIL))))))

(defun return_square (lst) (square_rec lst ()))
```

Результат работы:
(return_square '(1 2 3 4)) → (1 4 9 16)