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 №6

по курсу «Функциональное и логическое программирование»

Студент Рунов К.А.

Группа ИУ7-64Б

Оценка (баллы)

Преподаватели Толпинская Н. Б., Строганов Ю. В.

2024 г.

```
; 1. Написать хвостовую рекурсивную функцию my-reverse, которая развернет верхний уровень своего списка-аргумента lst
(defun my-reverse* (lst res)
  (cond ((null lst) res)
        (t (my-reverse* (cdr lst) (cons (car lst) res)))))

(defun my-reverse (lst)
  (my-reverse* lst nil))
```

```
; 3. Написать функцию, которая возвращает первый элемент списка-аргумента, который сам является непустым списком
(defun f3 (lst)
  (cond ((null lst) res)
        ((consp (car lst)) (car lst))
        (t (f3 (cdr lst)))))
```

```
; 4. Написать функцию, которая выбирает из заданного списка только те числа, которые больше 1 и меньше 10 (Вариант: между двумя заданными границами)
(defun f4* (lst from to res)
  (cond ((null lst) res)
        ((and (numberp (car lst))
              (> (car lst) from)
              (< (car lst) to))
         (f4* (cdr lst) from to (cons (car lst) res)))
        (t (f4* (cdr lst) from to res))))

(defun f4 (lst)
  (my-reverse (f4* lst 1 10 nil)))
```

```
; 5. Напишите рекурсивную функцию, которая умножает на заданное число-аргумент все числа из заданного списка-аргумента, когда
; а) все элемента списка - числа,
; б) элементы списка - любые объекты.
(defun all-numbersp (lst)
  (cond ((null lst) nil)
        ((and (numberp (car lst)) (null (cdr lst))) t)
        ((not (numberp (car lst))) nil)
        (t (all-numbersp (cdr lst)))))

```

```
(defun f5a (lst n)
  (cond ((all-numbersp lst) (mapcar #'(lambda (x) (* x n)) lst))
        (t lst)))

(defun apply-if-number (a fn)
  (cond ((numberp a) (apply fn (cons a nil)))
        (t a)))

(defun self (x) x)

(defun f5b (lst n)
  (mapcar #'(lambda (x)
              (apply-if-number x #'(lambda (y) (* y n)))) lst))
```

```
; 8. Написать рекурсивную версию (с именем rec-add) вычисления сумы чисел заданного списка:
; а) одноуровневого смешанного,
; б) 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))))
        ((consp (car lst))
         (+ sum (rec-add-b* (car lst) 0) (rec-add-b* (cdr lst) 0)))
        (t (rec-add-b* (cdr lst) sum))) ; meeeh

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

```
; 9. Написать рекурсивную версию с именем recnth функции nth.
(defun recnth (n lst)
  (cond ((and (numberp n) (= n 0)) (car lst))
```

```
4         ((and (numberp n) (> n 0)) (recnth (- n 1) (cdr lst)))
5         (t nil)))
```

```
1 ; 10. Написать рекурсивную функцию allodd, которая возвращает t, к
    огда все элементы списка нечетные.
2 (defun allodd (lst)
3   (cond ((not (numberp (car lst))) nil)
4         ((and (oddp (car lst)) (null (cdr lst))) t)
5         ((oddp (car lst)) (allodd (cdr lst)))
6         (t nil)))
```

```
1 ; 11. Написать рекурсивную функцию, которая возвращает первое нече
    тное число из списка (структурированного), возможно создавая не
    которые вспомогательные функции.
2 (defun first-odd (lst)
3   (cond ((and (numberp lst) (oddp lst)) lst)
4         ((atom lst) nil)
5         (t (or (first-odd (car lst))
6                (first-odd (cdr lst))))))
```

```
1 ; 12. Используя cons-дополняемую рекурсию с одним тестом завершени
    я, написать функцию, которая получает как аргумент список чисел
    , а возвращает список квадратов этих чисел в том же порядке.
2 (defun f12* (lst reslst)
3   (cond ((null lst) reslst)
4         (t (f12* (cdr lst) (cons (* (car lst) (car lst))
             reslst)))))
5
6 (defun f12 (lst)
7   (my-reverse (f12* lst nil)))
```