



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"

Тема Рекурсивные функции

Студент Цветков И.А.

Группа ИУ7-63Б

Оценка (баллы) _____

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

Москва — 2022 г.

1 Практические задания

1.1 Задание 1

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

```
1 (defun move-to (lst result)
2   (cond ((null lst) result)
3         (T (move-to (cdr lst) (cons (car lst) result))))
4   )
5 )
6
7 (defun my-reverse (lst)
8   (move-to lst ()))
9 )
```

1.2 Задание 2

Условие: написать функцию, которая возвращает первый элемент списка-аргумента, который сам является непустым списком.

```
1 (defun first-lst (lst)
2   (cond ((null lst) Nil)
3         ((and (not (null (car lst)))
4               (listp (car lst)))) (caar lst))
5         (T (first-lst (cdr lst)))
6   )
7 )
```

1.3 Задание 3

Условие: написать функцию, которая выбирает из заданного списка только те числа, которые больше 1 и меньше 10.

```

1 (defun check-border (num a b)
2   (or (> a num b) (< a num b))
3 )
4
5 (defun select-between-rec (lst a b result)
6   (cond ((null lst) result)
7         ((and (numberp (car lst))
8               (check-border (car lst) a b))
9          (select-between-rec (cdr lst) a b (cons (car
10            lst) result)))
11         (T (select-between-rec (cdr lst) a b result)))
12 )
13
14 (defun select-between (lst)
15   (select-between-rec lst 1 10 ()))
16 )

```

1.4 Задание 4

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

- все элементы списка — числа;
- элементы списка — любые объекты.

```

1 (defun move-to (lst result)
2   (cond ((null lst) result)
3         (T (move-to (cdr lst) (cons (car lst) result))))
4   )
5 )
6
7 (defun my-reverse (lst)
8   (move-to lst ()))
9 )
10
11 ; a)
12 (defun mul-num-rec (lst num result)

```

```

13      (cond ((null lst) result)
14             (T (mul-num-rec (cdr lst) num (cons (* (car lst) num)
15                                                    result))))
16    )
17
18 (defun mul-num (lst num)
19     (my-reverse (mul-num-rec lst num ())))
20 )
21
22 ; 6)
23 (defun mul-num-rec (lst num result)
24     (cond ((null lst) result)
25            ((symbolp (car lst))
26             (mul-num-rec (cdr lst) num (cons (car lst) result)))
27            ((numberp (car lst))
28             (mul-num-rec (cdr lst) num (cons (* (car lst) num)
29                                                result))))
29     (T (mul-num-rec (cdr lst) num
30                     (cons (my-reverse (mul-num-rec (car lst) num ()))
31                           result))))
31 )
32 )
33
34 (defun mul-num (lst num)
35     (my-reverse (mul-num-rec lst num ())))
36 )

```

1.5 Задание 5

Условие: 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)).

```

1 (defun check-border (num a b)
2     (or (> a num b) (< a num b))
3 )
4

```

```

5 (defun select-between-rec (lst a b result)
6   (cond ((null lst) result)
7         ((and (numberp (car lst))
8               (check-border (car lst) a b))
9           (select-between-rec (cdr lst) a b (cons (car
10              lst) result)))
11         (T (select-between-rec (cdr lst) a b result)))
12 )
13
14 (defun select-between (lst a b)
15   (select-between-rec lst a b ()))
16 )

```

1.6 Задание 6

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

- одноуровневого смешанного;
- структурированного.

```

1 ; a)
2 (defun rec-add-rec (lst result)
3   (cond ((null lst) result)
4         ((numberp (car lst)) (rec-add-rec (cdr lst) (+ (car lst)
5              result)))
6         (T (rec-add-rec (cdr lst) result)))
7   )
8
9 (defun rec-add (lst)
10   (rec-add-rec lst 0))
11 )
12
13 ; 6)
14 (defun rec-add-rec (lst result)
15   (cond ((null lst) result)

```

```

16      ((symbolp (car lst))
17        (rec-add-rec (cdr lst) result))
18      ((numberp (car lst))
19        (rec-add-rec (cdr lst) (+ (car lst) result)))
20      (T (rec-add-rec (cdr lst) (rec-add-rec (car lst) result)))
21    )
22  )
23
24 (defun rec-add (lst)
25   (rec-add-rec lst 0)
26 )

```

1.7 Задание 7

Условие: написать рекурсивную версию с именем `recnth` функции `nth`.

```

1 (defun recnth (lst n)
2   (cond ((null lst) Nil)
3         ((= n 0) (car lst))
4         (T (recnth (cdr lst) (- n 1))))
5   )
6 )

```

1.8 Задание 8

Условие: написать рекурсивную функцию `alldd`, которая возвращает `t`, когда все элементы списка нечетные.

```

1 (defun alldd (lst)
2   (cond ((null lst) T)
3         ((oddp (car lst)) (alldd (cdr lst)))
4         (T Nil))
5   )
6 )

```

1.9 Задание 9

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

```
1 (defun first-odd (lst)
2   (cond ((null lst) Nil)
3         ((and (numberp (car lst))
4               (oddp (car lst))) (car lst))
5         ((listp (car lst))
6          (or (first-odd (car lst)) (first-odd (cdr lst))))
7         (T (first-odd (cdr lst))))
8   )
9 )
```

1.10 Задание 10

Условие: 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.

```
1 (defun quad (lst)
2   (cond ((null lst) Nil)
3         (T (cons (* (car lst) (car lst)) (quad (cdr lst)) ))
4   )
5 )
```