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`.

```
(defun move-to (lst result)
    (cond ((null lst) result)
            (T (move-to (cdr lst) (cons (car lst) result)))
    )
)

(defun my-reverse (lst)
    (move-to lst ())
)
```

1.2 Задание 2

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

```
(defun first-lst (lst)
    (cond ((null lst) Nil)
          ((and (listp (car lst)) (not (null (car lst)))) (caar
             lst))
          (T (first-lst (cdr lst)))
    )
)
```

1.3 Задание 3

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

```
(defun check-border (num a b)
    (or (> a num b) (< a num b))
)

(defun select-between-rec (lst a b result)
    (cond ((null lst) result)
          ((and (numberp (car lst))
              (check-border (car lst) a b))
                  (select-between-rec (cdr lst) a b (cons (car
                     lst) result)))
          (T (select-between-rec (cdr lst) a b result))
    )
)

(defun select-between (lst)
    (select-between-rec lst 1 10 ())
)
```

1.4 Задание 4

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

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

```
(defun move-to (lst result)
    (cond ((null lst) result)
          (T (move-to (cdr lst) (cons (car lst) result)))
    )
)

(defun my-reverse (lst)
    (move-to lst ())
)

; a)
(defun mul-num-rec (lst num result)
```

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

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
                        lst) result)))
10          (T (select-between-rec (cdr lst) a b result))
11      )
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:

- одноуровнего 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) result)))
5             (T (rec-add-rec (cdr lst) result))
6       )
7   )
8
9
10  (defun rec-add (lst)
11      (rec-add-rec lst 0)
12  )
13
14
15  ; б)
16  (defun rec-add-rec (lst result)
17      (cond ((null lst) result)
```

```
18          ((listp (car lst)) (rec-add-rec (cdr lst) (rec-add-rec
               (car lst) result)))
19          ((numberp (car lst)) (rec-add-rec (cdr lst) (+ (car lst)
               result)))
20          (T (rec-add-rec (cdr 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 `allodd`, которая возвращает `t`, когда все элементы списка нечетные.

```
1  (defun allodd (lst)
2      (cond ((null lst) T)
3            ((oddp (car lst)) (allodd (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.

```
(defun first-odd (lst)
    (cond ((null lst) Nil)
          ((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)))
    )
)
```

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.

```
(defun quad (lst)
    (cond ((null lst) Nil)
          (T (cons (* (car lst) (car lst)) (quad (cdr lst)) ))
    )
)
```