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 №4

по курсу «Функциональное и логическое программирование»

на тему: «Использование управляющих структур, работа со списками»

Студент ИУ7-61Б _____ Савинова М. Г.
(Группа) (Подпись, дата) (Фамилия И. О.)

Преподаватель _____ Толпинская Н. Б.
(Подпись, дата) (Фамилия И. О.)

Преподаватель _____ Строганов Ю.В.
(Подпись, дата) (Фамилия И. О.)

*2024 г.*

# СОДЕРЖАНИЕ

# 1 Практические задания

## 1.1 Задание 1

Чем принципиально отличаются функции `cons`, `list`, `append`?
**Ответ:**

— `cons` — базовая функция, которая объединяет значения двух своих аргументов в точечную пару.

— `list` — принимает произвольное число аргументов и возвращает список, состоящий из значений аргументов.

— `append` — объединяет списки в один, при этом оригинальные списки остаются нетронутыми, и возвращается новый список, содержащий комбинированные элементы.

Каковы результаты вычисления следующих выражений?

```
(setf lst1 '(a b c))
(setf lst2 '(d e))

(print (cons lst1 lst2))    ;; ((A B C) D E)
(print (list lst1 lst2))    ;; ((A B C) (D E))
(print (append lst1 lst2)) ;; (A B C D E)
```

## 1.2 Задание 2

Каковы результаты вычисления следующих выражений, и почему?

```
(print (reverse '(a b c)))        ;; (C B A)
(print (reverse '(a b (c (d))))) ;; ((C (D)) B A)
(print (reverse '(a)))            ;; (A)
(print (reverse ()))              ;; NIL
(print (reverse '((a b c))))      ;; ((A B C))

(print (last '(a b c)))           ;; (C)
(print (last '(a)))               ;; (A)
(print (last '(a b (c))))         ;; ((C))
(print (last ()))                 ;; NIL
```

**Ответ:**

— `reverse` работает не разрушающим образом и не изменяет исходный список, а возвращает новый список с элементами в обратном порядке. Работает только с элементами верхнего уровня;

— `last` используется для извлечения последнего элемента из списка. Она принимает один аргумент — список и возвращает последний элемент этого списка. Работает только с элементами верхнего уровня;

## 1.3 Задание 3

Написать два варианта функции, которая возвращает последний элемент своего списка-аргумента.

```
(defun last_1 (lst)
    (last lst)
)

(defun last_2 (lst)
    (cond ((null lst) Nil)
          (t (cons (car (reverse lst)) Nil))
    )
)

(defun last_3 (lst)
    (cond ((cdr lst) (last_3 (cdr lst)))
          ((cond ((null lst) Nil)
                 (t (cons (car lst) Nil)))
          )
    )
)
```

## 1.4 Задание 4

Написать два варианта функции, которая возвращает свой список аргумента без последнего элемента.

```
(defun no_last_1 (lst)
    (reverse (cdr (reverse lst)))
)

```

```
(defun no_last_2 (lst)
    (cond ((null (cdr lst)) Nil)
          (t (cons (car lst) (no_last_2 (cdr lst)))))
)
```

## 1.5 Задание 5

Написать функцию swap-first-last, которая переставляет в списке-аргументе первый и последний элемент.

```

(defun get_middle (lst x)
    (cond ((null lst) lst)
          ((null (cdr lst)) (cons x Nil))
          (t (cons (car lst) (get_middle (cdr lst) x)))
    )
)

(defun swap (lst)
    (cons (car (last lst)) (get_middle (cdr lst) (car lst)))
)
```

## 1.6 Задание 6

Написать простой вариант игры в кости, в котором бросается две правильные кости. Если сумма выпавших очков равна 7 или 11 — выигрыш, если выпало (1, 1) или (6, 6) — игрок имеет право снова бросить кости, во всех остальных случаях ход переходит ко второму игроку, но запоминается сумма выпавших очков. Если второй игрок не выигрывает абсолютно, то выигрывает тот игрок, у которого больше очков. Результаты игры и значения выпавших костей выводить на экран с помощью `print`.

```
(defun get_sum (dice)
    (+ (car dice) (cdr dice))
)

(defun roll_dice ()
    (setf *random-state* (make-random-state t))
    (cons (+ (random 5) 1)
          (+ (random 5) 1)
    )
```

```
)

(defun is_abs_win (dice)
    (let (
            (sum (get_sum dice))
        )
        (or (= sum 7) (= sum 11))
    )
)

(defun is_reroll (dice)
    (let (
            (fir (car dice))
            (sec (cdr dice))
        )
        (or (and (= fir 6) (= sec 6))
            (and (= fir 1) (= sec 1)))
    )
)

(defun get_dice_res (id)
    (let (
            (dices (roll_dice))
        )
        (print 'Player)
        (princ id)
        (print 'dices)
        (princ dices)
        (cond ((is_reroll dices)
               (print 'Rerolling)
               (get_dice_res id))
              (t dices))
    )
)

(defun game ()
    (let (
            (dice1 (get_dice_res 1))
            (dice2 (get_dice_res 2))
        )
        (cond (
```

```
                    (is_abs_win dice1)
                    (print "Player1 won absolutely")
                )
                (
                    (is_abs_win dice2)
                    (print "Player2 won absolutely")
                )
                (
                    (> (get_sum dice1) (get_sum dice2))
                    (print "Player1 won")
                )
                (
                    (< (get_sum dice1) (get_sum dice2))
                    (print "Player2 won")
                )
                (
                    (= (get_sum dice1) (get_sum dice2))
                    (print "Draw")
                )
            )
    )
)

(game)
```

## 1.7 Задание 7

Написать функцию, которая по своему списку-аргументу `lst` определяет является ли он палиндромом.

```
(defun pal (lst)
    (equal lst (reverse lst))
)
```

## 1.8 Задание 8

Напишите **свои** необходимые функции, которые обрабатывают таблицу из 4-х точечных пар: (страна . столица), и возвращает по стране — столицу, а по столице — страну.

```
(defun both (tbl name)
    (cond ((null tbl) Nil)
```

```
3           ((eql name (cdar tbl)) (caar tbl))
4           ((eql name (caar tbl)) (cdar tbl))
5           (t (f (cdr tbl) name))
6       )
7   )
8
9   (defun con_by_cap (tbl name)
10      (cond ((null tbl) Nil)
11          ((eql name (caar tbl)) (cdar tbl))
12          (t (f1 (cdr tbl) name))
13      )
14  )
15
16  (defun cap_by_con (tbl name)
17      (cond ((null tbl) Nil)
18          ((eql name (cdar tbl)) (caar tbl))
19          (t (f2 (cdr tbl) name))
20      )
21  )
```

## 1.9 Задание 9

Написать функцию, которая умножает на заданное число-аргумент первый числовой элемент списка из заданного 3-х элементного списка-аргумента, когда

1) все элементы списка — числа;

2) элементы списка — любые объекты.

```
1   (defun mul (lst x)
2       (cond ((null lst) Nil)
3           ((numberp (car lst)) (cons (* (car lst) x) (cdr lst)))
4           (t (cons (car lst) (mul (cdr lst) x)))
5       )
6   )
```