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, искусственный интеллект и системы управления**»

КАФЕДРА «**Программное обеспечение ЭВМ и информационные технологии**»

Отчет по лабораторной работе №4

по курсу «Функциональное и логическое программирование»

«Использование управляющих структур, работа со списками»

Студент группы **ИУ7-64Б** _____ **Д.С. Чепиго**

(Подпись, дата) (И.О. Фамилия)

Преподаватели _____ **Н.Б. Толпинская**

(Подпись, дата) (И.О. Фамилия)

_____ **Ю.В. Строганов**

(Подпись, дата) (И.О. Фамилия)

***2023** г.*

Содержание

1 Практические задания

1.1 Задание №1

Пусть (setf lst1 ’(a b c))

(setf lst2 ’(d e))

Каковы результаты вычисления следующих выражений?

1. (cons lstl lst2)

   ((A B C) D E)

2. (list lst1 lst2)

   ((A B C) (D E))

3. (append lst1 lst2)

   (A B C D E)

1.2 Задание №2

Каковы результаты вычисления следующих выражений, и почему?

1) (reverse ’(a b c))

   (C B A) – обычный разворот списка

2) (reverse ’(a b (c (d))))

   ((C (D)) B A) – так как reverse работает относительно элементов списка верхнего уровня

3) (reverse ’(a))

   (A)

4) (reverse ())

   NIL

5) (reverse ’((a b c)))

   ((A B C)) – так как reverse работает относительно элементов списка верхнего уровня, а (A B C) единственный элемент, то он останется на своем месте

6) (last ’(a b c))

   (C) – так как является последним элементом

7) (last '(a))

   (A) – так как является единственным, то есть и первым и последним элементом

8) (last '((a b c)))

   ((A B C)) – так как last работает относительно элементов списка верхнего уровня, а (A B C) единственный элемент, то есть и первый и последний

9) (last '(a b (c)))

   ((C)) – последний элемент списка верхнего уровня

10) (last ())

    (NIL)

1.3 Задание №3

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

```
(defun lastVar1 (x)
    (car (reverse x)))

(defun lastVar2 (x)
    (if (null (cdr x))
        (car x)
        (lastVar2 (cdr x))))
```

1.4 Задание №4

Написать, по крайней мере, два варианта функции, которая возвращает свой список аргумент без последнего элемента.

```
(defun withoutLast1 (x)
    (reverse (cdr (reverse x))))

(defun withoutLast2 (x)
    (if (null (cdr x))
```

```
6           NIL
7           (cons (car x) (withoutLast2 (cdr x)))))
```

1.5 Задание №5

Напишите функцию swap-first-last, которая переставляет в списке- аргументе первый и последний элементы.

```
1  (defun change-last (x el)
2      (cond ((null x) x)
3      ((null (cdr x)) (cons el Nil))
4      (T (cons (car x) (change-last (cdr x) el)))))
5
6  (defun changeLast (x el)
7      (if (null x)
8          (x)
9          (if (null (cdr x))
10             (cons el Nil)
11             (cons (car x) (changeLast (cdr x) el)))))
12
13 (defun swap-first-last (x)
14     (changeLast (cons (car (myLast x)) (cdr x)) (car x)))
```

1.6 Задание №6

Написать простой вариант игры в кости, в котором бросаются 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. Результат игры и значения выпавших костей выводить на экран с помощью функции print.

```
(defun roll_dice ()
    ( setf *random-state* (make-random-state t))
    (+ (random 6) 1))

(defun check_continue_game (result)
    (not (or (= result 7) (= result 11))) )

(defun make_a_move (player_i)
    (let ((dice1 (roll_dice)) (dice2 (roll_dice)))

    (if (and
        (print (list 'Игрок player_i 'бросает 'кости))
        (= dice1 dice2) (or (= dice1 1) (= dice1 6)))
        (and
            (print (list 'Выпало dice1 dice2 'повтор))
            (make_a_move player_i )
        )
        (and
            (print (list 'Выпало dice1 dice2))
            (+ dice1 dice2)))))

(defun compare_results (res1 res2)
    (if (check_continue_game res2)
    (and
        (print (list 'Сравнение 'по 'очкам))
        (print ( list 'Игрок 1 'набрал res1))
        (print ( list 'Игрок 2 'набрал res2))
        (cond
            ((< res1 res2)
            (and
            (print (list 'Игрок 2 'выиграл 'по 'очкам))
            2 ))

```

```
            ((> res1 res2)
            (and
            (print (list 'Игрок 1 'выиграл 'по 'очкам))
            1 ))
            ((and (print 'Ничья ()) 0))))

    (and
        (print (list 'Игрок 2 'набрал res2 'очков
        'и 'выиграл 'абсолютно)) 2 )))

(defun game ()
    (let ((res1 (make_a_move 1)))
        (if (check_continue_game res1)
            (compare_results res1 (make_a_move 2))
            (and (print (list 'Игрок 1 'набрал res1 'очков
            'и 'выиграл 'абсолютно)) 1))))

(game)
```

1.7 Задание №7

Написать функцию, которая по своему списку-аргументу lst определяет является ли он палиндромом (то есть равны ли lst и (reverse lst)).

```
(defun listEquale(x)
    (equal x (reverse x)))
```

1.8 Задание №8

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

```
(defun countryByCapital (table capital)
    (if (equal capital (cdr (car table)))
```

```
        (car (car table))
        (countryByCapital (cdr table) capital)))

(defun capitalByCountry (table country)
    (if (equal country (car (car table)))
        (cdr (car table))
        (capitalByCountry (cdr table) country)))
```

1.9 Задание №9

Напишите функцию, которая умножает на заданное число-аргумент первый числовой элемент списка из заданного 3-х элементного списка-аргумента, когда все элементы списка – числа/элементы списка – любые объекты.

```
(defun mulFirstNumbers (lst number)
    (cons (* number (car lst)) (cdr lst)))

(defun mulFirstAny (lst number)
    (cond ((null lst)   lst)

          ((numberp (car lst))
                    (cons (* (car lst) number) (cdr lst)))

          (T (cons (car lst)
                   (mulFirstAny (cdr lst) number)))))
```