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 дисциплине "Функциональное и логическое программирование"

Студент Кузнецова А. В.

Группа ИУ7-61Б

Оценка (баллы)

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

Москва — 2023 г.

1 Практические задания

1.1 Задание №1

Чем принципиально отличаются функции cons, list, append?

Пусть

```
1  (setf lst1 '( a b c))
2  (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)
2  (reverse '(a b (c (d)))) => ((c (d)) b a)
3  (reverse '(a)) => (a)
4  (last '(a b c)) => (c)
5  (last '(a)) => (a)
6  (last '((a b c))) => ((a b c))
7  (reverse ()) => Nil
8  (reverse '((a b c))) => ((a b c))
9  (last '(a b (c))) => ((c))
10 (last ()) => Nil
```

1.3 Задание №3

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

```
(defun get-last1 (lst)
    (car (last lst))

(defun get-last2 (lst)
    (if (cdr lst)
        (get-last2 (cdr lst))
        (car lst)))

(defun get-last3 (lst)
    (car (reverse lst)))
```

1.4 Задание №4

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

```
(defun get-no-last1 (lst)
    (reverse (cdr (reverse lst))))

(defun get-no-last2 (lst)
    (if (cdr lst)
        (cons (car lst) (get-no-last2 (cdr lst)))))
```

1.5 Задание №5

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

```
(defun swap-first-last (lst)
    (cond ((null (cdr lst)) lst)
        (T (cons (car (last lst))
            (reverse (cons (car lst) (cdr (reverse (cdr
               lst)))))))))
```

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.

```
1     (setf *random-state* (make-random-state T))
2
3     (defun roll-dice ()
4         (list (+ (random 6) 1) (+ (random 6) 1)))
5
6     (defun dice-sum (dice)
7         (apply #'+ dice))
8
9     (defun check-win (sum)
10        (or (= sum 7) (= sum 11)))
11
12    (defun check-repeat (dice)
13        (let ((f (car dice))
14              (s (cadr dice)))
15            (or (= f s 1) (= f s 6))))
16
17    (defun player-round (who)
18        (let* ((player-dice (roll-dice))
19               (player-sum (dice-sum player-dice)))
20              (print who)
21              (print player-dice)
22              (cond ((check-win player-sum) Nil)
23                    ((check-repeat player-dice) (player-round
                        who))
24                    (T player-sum))))
25
26    (defun dice ()
27        (let ((first-sum (player-round "first")))
28            (if (not first-sum)
29                (print "First␣player␣is␣winner!")
```

```
30                  (let ((second-sum (player-round "second")))
31                      (cond ((or (not second-sum) (< first-sum
                            second-sum))
32                                 (print "Second player is winner!"))
33                            ((> first-sum second-sum)
34                                 (print "First player is winner!"))
35                            (T (print "Draw."))))))
36
37 (dice)
```

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)).

```
1     (defun is-palindrome (lst)
2         (equal lst (reverse lst)))
```

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.

```
1     (defun get-capital (table country)
2         (cond ((null table) Nil)
3             ((equal (caar table) country) (cdar table))
4             (T (get-capital (cdr table) country))))
5
6     (defun get-country (table capital)
7         (cond ((null table) Nil)
8             ((equal (cdar table) capital) (caar table))
9             (T (get-country (cdr table) capital))))
```

1.9 Задание №9

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

а) все элементы списка — числа

```
1  (defun mul-first-nums (lst num)
2      (and lst (cons (* (car lst) num) (cdr lst))))
```

б) элементы списка – любые объекты

```
1  (defun mul-first-num (lst num)
2      (cond ((null lst) lst)
3          ((numberp (car lst)) (cons (* (car lst) num) (cdr
               lst)))
4          (T (cons (car lst) (mul-first-num (cdr lst) num)))))
```