**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»**

**Тема** Использование управляющих структур, работа со списками

**Студент** Климов И.С.

**Группа** ИУ7-62Б

**Оценка (баллы)**

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

Москва — 2022 г.

## **Задание 1**

Чем принципиально отличаются функции cons, list, append?

1. cons – помещает первый аргумент в начало второго;
2. list – создает список, состоящий из аргументов;
3. append – создает список, состоящий из элементов аргументов.

Пусть (setf lst1 '(a b))

(setf lst2 '(c d)).

Каковы результаты вычисления следующих выражений?

1. (cons lstl lst2) -> ((A B) C D)
2. (list lst1 lst2) -> ((A B) (C D))
3. (append lst1 lst2) -> (A B C D)

## **Задание 2**

Каковы результаты вычисления следующих выражений, и почему?

1. (reverse ()) -> NIL
2. (last ()) -> NIL
3. (reverse '(a)) -> (A)
4. (last '(a)) -> (A)
5. (reverse '((a b c))) -> (C B A)
6. (last '((a b c))) -> ((A B C))

## **Задание 3**

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

**Решение**

(defun last\_v1 (x)

(if (listp x)

(if (<= (length x) 1)

(car x)

(last\_1 (cdr x)))

NIL))

(defun last\_v2 (x)

(cond ((null (listp x)) NIL)

((null (cdr x)) (car x))

(T (last\_v2 (cdr x)))))

(defun last\_v3 (x)

(if (or (null (listp x)) (null x))

NIL

(nth (- (length x) 1) x)))

(defun last\_v4 (x)

(if (listp x)

(car (reverse x))

NIL))

## **Задание 4**

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

**Решение**

(defun except\_last\_v1 (x)

(if (null (and (listp x) (cdr x)))

NIL

(cons (car x) (except\_last\_v1 (cdr x)))))

(defun except\_last\_v2 (x)

(if (listp x)

(reverse (cdr (reverse x)))))

(defun except\_last\_v3 (x)

(if (listp x)

(butlast x)

NIL))

## **Задание 5**

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.

**Решение**

(defun roll\_dice (edges)

(let ((dice\_1 (random edges))

(dice\_2 (random edges)))

(let ((sum (+ dice\_1 dice\_2)))

(or (format T "Dropped dice = (~a ~a), sum = ~a~%" dice\_1 dice\_2 sum)

(cond ((or (= sum 7) (= sum 11))

(or (format T "~%") T))

((or (= 1 dice\_1 dice\_2) (= 6 dice\_1 dice\_2))

(or (format T "Player rolls the dice again~%")

(setq sum (+ sum (roll\_dice edges)))))

(T (or (format T "~%") sum)))))))

(defun roll\_player\_dice (player)

(or (format T "Player ~a rolls dice~%" player)

(roll\_dice 6)))

(defun get\_winner (player\_1 player\_2)

(cond ((equal player\_1 T) 1)

((equal player\_2 T) 2)

((> player\_1 player\_2) 1)

((> player\_2 player\_1) 2)

(T 0)))

(defun play\_dice ()

(let ((winner (get\_winner (roll\_player\_dice 1) (roll\_player\_dice 2))))

(if (= winner 0)

(format T "The result of the game is draw~%")

(format T "The winner is player ~a~%" winner))))