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 _____ Мицевич М. Д.
(Группа) (Подпись, дата) (И. О. Фамилия)

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

2022 г.

Задание 1

Постановка задачи

Чем принципиально отличаются функции `cons`, `list`, `append`?

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

Каковы результаты следующих выражений?

```
(cons lst1 lst2)
(list lst1 lst2)
(append lst1 lst2)
```

Решение

1. `((A B) C D)`
2. `((A B) (C D))`
3. `(A B C D)`

Задание №2

Постановка задачи

Каковы результаты вычисления следующих выражений?

```
(reverse ())
(last ())
(reverse '(a))
(last '(a))
(reverse '((a b c)))
(last '((a b c)))
```

Решение

1. `Nil`
2. `Nil`
3. `(a)`
4. `(a)`

5. `((a b c))`

6. `((a b c))`

Задание №3

Постановка задачи

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

Решение

Листинг 1 – Решение задания №3

```
(defun last-1 (lst)
    (if (eql (cdr lst) nil)
        (car lst)
        (last-1 (cdr lst))))

(defun last-2 (lst)
    (reduce #'(lambda (prev next) next) lst))
```

Задание №4

Постановка задачи

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

Решение

Листинг 2 – Решение задания №3

```
(defun rem-last-2-arg (lst res)
    (if (cdr lst)
        (rem-last-2-arg (cdr lst) (append res (list (car lst))))
        res))

(defun rem-last-1 (lst)
    (rem-last-2-arg lst ()))
```

```
(defun rem-last-2 (lst)
    (cond ((null lst) nil)
          ((null (cdr lst)) nil)
          (t (cons (car lst) (rem-last-2 (cdr lst))))))
```

Задание №5

Постановка задачи

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 получает 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

Листинг 3 – Решение задания №3

```
(defun is-win-comb (p1 p2)
    (or (eql (+ p1 p2) 7) (eql (+ p1 p2) 11)))

(defun is-repeat (p1 p2)
    (or (and (eql p1 1) (eql p2 1)) (and (eql p1 6) (eql p2 6))))

(defun play-step ()
    (let* ((cube1p (+ (random 6) 1))
           (cube2p (+ (random 6) 1)))
        (cond ((is-win-comb cube1p cube2p)
               (and (print `(wining combination cube1 - ,cube1p
                  cube2 - ,cube2p))
                    (list t (+ cube1p cube2p))))
              ((is-repeat cube1p cube2p) (play-step))
              (t
               (and (print `(not wining combination cube1 - ,
                  cube1p cube2 - ,cube2p))
                    (list nil (+ cube1p cube2p)))))))

(defun play-game ()
```

```
(let ((res1 (and (print '(player1 step)) (play-step))))
    (if (first res1)
        (print '(player1 is a winner))
        (let ((res2 (and (print '(player2 step)) (play-step))
           ))
            (if (or (first res2) (> (second res2) (second
               res1)))
                (print '(player2 is a winner))
                (if (eql (second res1) (second res2))
                    (print '(DRAW))
                    (print '(player1 is a winner)))))))
```

Контрольные вопросы

Вопрос 1. Синтаксическая форма и хранение программы в памяти.

Ответ. В Lisp формы представления программы и обрабатываемых ею данных одинаковы – они представлены в виде S-выражений. Программы могут обрабатывать и преобразовывать другие программы или сами себя. В памяти программа представляется в виде бинарных узлов, так как она состоит из S-выражений.

Вопрос 2. Трактовка элементов списка.

Ответ. Если отсутствует блокировка вычислений, то первый элемент списка трактуется как имя функции, а остальные элементы – как аргументы функции.

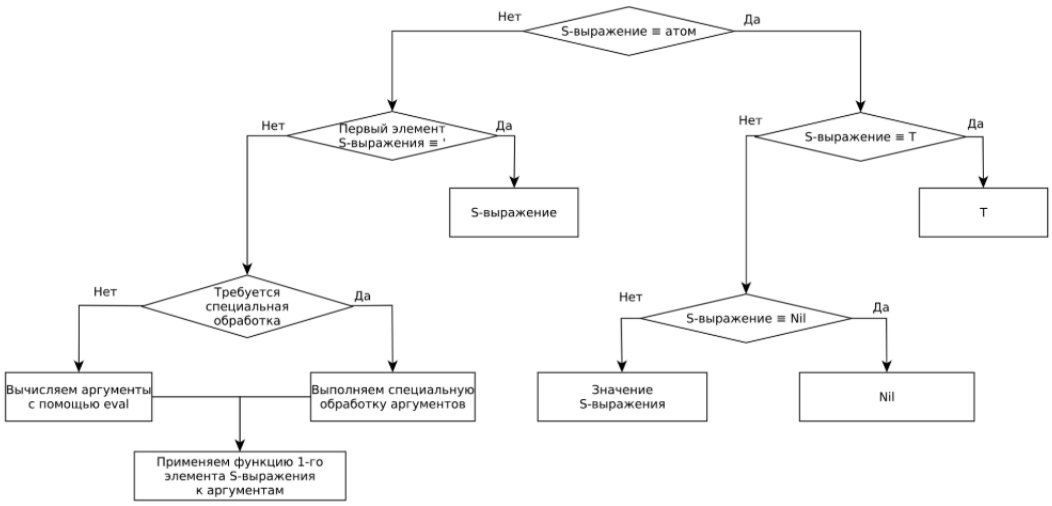

Рисунок 1 – Схема работы функции **eval**.

Вопрос 3. Порядок реализации программы.

Ответ. Работа программы циклична: сначала программа ожидает ввода S-выражения, затем передает полученное S-выражение интерпретатору – функции eval, а в конце, после отработки функции eval, выводит последний полученный результат.

Вопрос 4. Способы определения функции.

Ответ. Функцию можно определить с помощью **defun** или **lambda.**
(defun имя_функции (список_аргументов) тело_функции)
(lambda (список_аргументов) тело_функции).