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»

Студент Маслова Марина Дмитриевна

Группа ИУ7-63Б

Оценка (баллы) _____

Преподаватель Толпинская Наталья Борисовна

Преподаватель Строганов Юрий Владимирович

2022 г.

1 Практические задания

1.1 Задание №1

Пусть

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

Каковы результаты вычисления следующих выражений?

```
(cons lst1 lst2)   ; ((a b) c d)
(list lst1 lst2)   ; ((a b) (c d))
(append lst1 lst2) ; (a b c d)
```

1.2 Задание №2

Каковы результаты вычисления следующих выражений, и почему?

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

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)
    (nreverse (cdr (reverse lst))))
```

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

1.5 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.

```
(setf *random-state* (make-random-state T))

(defun roll-dice ()
    (list (+ (random 6) 1) (+ (random 6) 1)))

(defun dice-sum (dice)
    (apply #'+ dice))

(defun check-win (sum)
    (or (= sum 7) (= sum 11)))

(defun check-repeat (dice)
    (let ((f (car dice))
          (s (cadr dice)))
          (or (= f s 1) (= f s 6))))

(defun player-round (who)
    (let* ((player-dice (roll-dice))
           (player-sum (dice-sum player-dice)))
          (print who)
          (print player-dice)
          (cond ((check-win player-sum) Nil)
                ((check-repeat player-dice) (player-round who))
                (T player-sum))))

(defun dice ()
    (let ((first-sum (player-round "first")))
          (if (not first-sum)
              (print "First player is winner!")
              (let ((second-sum (player-round "second")))
                   (cond ((or (not second-sum) (< first-sum second-sum))
                              (print "Second player is winner!"))
                         ((> first-sum second-sum)
                              (print "First player is winner!"))
                         (T (print "Draw."))))))))

(dice)
```

2 Теоретические вопросы

2.1 Синтаксическая форма и хранение программы в памяти

В Lisp и программа, и данные представлены S-выражениями, благодаря чему программа может обрабатывать и преобразовывать другие программы или саму себя. S-выражение — атом или точечная пара. Атом представляется в памяти 5-ю указателями: имя, значение, функция, свойство, пакет. Точечная пара представляется в памяти списковой ячейкой: бинарный узел, состоящий из двух указателей (car-указатель и cdr-указатель).

2.2 Трактовка элементов списка

По умолчанию список является формой (вычислимым выражением), в которой первый элемент трактуется как имя фунции, остальные элементы — как ее аргументы. Для возможности различия программы от данных создана функция quote и ее сокращенное обозначение — апостроф `'`. Функция quote и апостроф блокируют вычисление своего аргумента и возращают его текстовую запись.

2.3 Порядок реализации программы

Программа на языке Lisp включает определения новых функций на базе встроенных функций и другий функций, определенных в этой программе, а также вызовы функций для конкретных значений аргументов. Последовательность операций достигается вызовом функций в определенном порядке, т. е. суперпозицией функций. Передача данных между функциями выполняется через их аргументы и возвращаемые значения.

Выполнение программы заключается в вычислении значений функций для конкретных значений аргументов. Вычисление функции происходит с помощью интерпретации — функции eval, которая может вычислять значение любой формы. Так как программа на языке Lisp представлет S-выражение функция eval принимает в качестве аргумента S-выражение и вычисляет по схеме, представленной на рисунке 2.1.

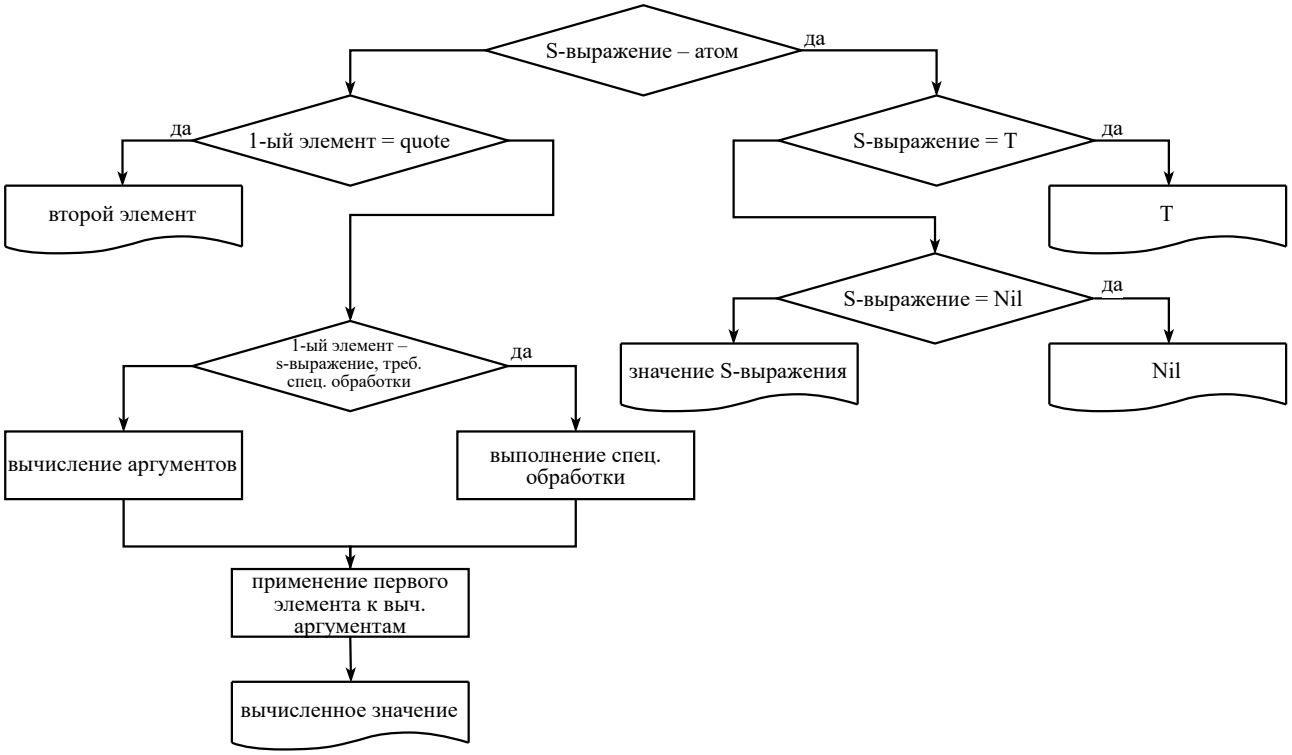

Рисунок 2.1 – Работа функции eval

2.4 Способы определения функции

Лямбда определение:

```
1 (lambda <lambda-список> <форма>) ; lambda-выражение
2 ;; <lambda-список> -- список аргументов
3 ;; <форма> -- тело функции
```

Определение функций с именем:

```
1 (defun <имя> <lambda-выражение>)
```