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»

Лабораторная работа № 7

Тема Функции более высокого порядка

Студент Сушина А.Д.

Группа ИУ7-61б

Оценка (баллы) _____

Преподаватель Толпинская Н.Б.

Москва.
2020 г

Цель работы — приобрести навыки использования функций более высокого порядка в Lisp

Задачи работы: изучить свойства символа, функциональное значение функций, способы композиции функций, способы организации и использования функций более высокого порядка в lisp

Задание 1

Чем принципиально отличаются функции cons, list, append?

Пусть (setf lst1 '(a b)), (setf Ist2 '(c d)). Каковы результаты вычисления следующих выражений?

1. (cons lst1 lst2)

2. (list lst1 Ist2)

3. (append Ist1 Ist2)

Функция cons создает точечную пару (устанавливает указатель car на первый полученный аргумент, cdr- на второй), принимает фиксированное число аргументов, равное двум.

Функция list создает список из своих аргументов. Принимает нефиксированное число аргументов. Написана на основе cons.

Функция append «сливает» свои аргументы в единый список. Обязательно требует, чтобы аргументы были списками.

1. ((A B) C D)

2. ((A B) (C D))

3. (A B C D)

Задание 2

Каковы результаты вычисления следующих выражений?

1. (reverse ()) → Nil

2. (last ()) → Nil

3. (reverse '(a)) → A

4. (last '(a)) → A

5. (reverse '((a b c))) → ((A B C))

6. (last '((a b c))) → ((A B C)

Задание 3

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

(defun get_last1 (lst) (car (reverse lst)))

(defun get_last2 (lst)

```
   (cond ((equal (cdr lst) nil) (car lst))
         (t (get_last2 (cdr lst)))))
```

Задание 4

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

```
(defun cut (lst) (reverse (cdr (reverse lst))))
```

```
(defun cut3 (lst rst)
   (cond ((null (cdr lst)) (cdr rst))
         (t (cut3 (cdr lst) (append rst (cons (car lst) nil))))))
```

```
(defun cut2 (lst)
   (cut3 lst '(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 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 ()  (list (+ (random 6) 1) (+ (random 6) 1)))
```

```
(defun play-sum (res)
   (+ (car res) (cadr res))
)
```

```
(defun is-win (res)
   (or (equal 7 (play-sum res))
      (equal 11 (play-sum res))
   )
```

```
)

(defun is-lucky (res)
   (or (equal res '(1 1))
      (equal res '(6 6))
   )
)

(defun play ()
   (setq player1 (roll-dice))
   (format T " First player ~A" player1)
   (cond ((equal T (is-win player1)) (print " First player win!"))
      ((equal T (is-lucky player1))(play))
      (t (last
         (list
            (setq player2 (roll-dice))
            (format T " Second player ~A" player2)
            (cond ((= (play-sum player1) (play-sum player2)) (print "Tie!"))
               ((> (play-sum player1) (play-sum player2)) (print "First player win!"))
               (t (print "Second player win!"))
            )
         )
      )
   )
```