

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический  
университет имени Н.Э. Баумана»  
(МГТУ им. Н.Э. Баумана)

---

---

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

---

**Лабораторная работа № 7**

<b>Дисциплина</b>	Функциональное и логическое программирование.
<b>Тема</b>	Работа с функционалом и рекурсией
<b>Студент</b>	Куприй А. А.
<b>Группа</b>	ИУ7-63Б
<b>Преподаватель</b>	Толпинская Н.Б., Строганов Ю.В.

Москва, 2020 г.

## 1 Практическая часть

### 1.1 Задание №1

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

```
1 (setf lst1 '(a b))
2 (setf lst2 '(c d))
3
4 (cons lst1 lst2)    ;;; ((A B) C D)
5 (list lst1 lst2)    ;;; ((A B) (C D))
6 (append lst1 lst2)  ;;; (A B C D)
```

Функция `cons` создает списковую ячейку и кладет в голову первый аргумент, а в хвост – второй. Функция `list` создает списковые ячейки для каждого аргумента и соединяет их в один список. А функция `append` объединяет два списка в один, состоящий из элементов двух списков.

### 1.2 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))
```

### 1.3 Задание №3

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

Рекурсивно – пока второй элемент не `Nil` идем дальше, иначе возвращаем голову:

```
1 (defun last_elem (lst)
2   (if (NULL (cadr lst)) (car lst)
3       (last_elem (cdr lst))
4   )
5 )
```

С использованием функционала – с использованием функционала `reduce`, возвращая последний полученный результат, а возвращаем каждый раз второй аргумент:

```
1 (defun last_elem (lst)
2   (reduce #'(lambda (a x) x) lst)
3 )
```

#### 1.4 Задание №4

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

С использованием функционала – пользуясь тем, что `mapcar` будет работать до хвоста в самом коротком списке (`cdr lst` всегда на один короче `lst`), возвращаем значение из `lst`, таким образом получится список без последнего элемента:

```
1 (defun centr (lst)
2   (mapcar (lambda (x y) y) (cdr lst) lst)
3 )
```

Рекурсивно – пока хвост не `Nil`, соединяем с помощью `cons` голову и возвращаем значение функции от хвоста:

```
1 (defun centr (lst)
2   (if (NULL (cadr lst)) ()
3       (cons (car lst) (centr (cdr lst)))
4   )
5 )
```

#### 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 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.

```
1 (defun print (x y s)
2   (print (list x '+ y '= s))
3 )
4
5 (defun analyze (x y)
6   (cond
7     ((and (= 1 x) (= 1 y)) (print x y (+ x y))
8       (analyze (+ 1 (random 6)) (+ 1 (random 6))))
9     ((and (= 6 x) (= 6 y)) (print x y (+ x y))
10      (analyze (+ 1 (random 6)) (+ 1 (random 6))))
11    (T (printS x y (+ x y)) (+ x y))
12   )
13 )
14
15 (defun main ()
16   (let ((s1 (analyze (+ 1 (random 6)) (+ 1 (random 6))))
17         (s2 (analyze (+ 1 (random 6)) (+ 1 (random 6)))))
18     (cond
19       ((or (= s1 7) (= s1 11) (> s1 s2)) 'first-win)
20       ((or (= s1 7) (= s2 11) (< s1 s2)) 'second-wins)
21       (T 'draw)
22     )
23   )
24 )
```