



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»

НАПРАВЛЕНИЕ ПОДГОТОВКИ _____ «09.03.04 Программная инженерия»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

Название: _____ Использование управляющих структур, работа со списками

Дисциплина: _____ Функциональное и логическое программирование

Студент	ИУ7-66Б	_____	А.Д. Ковель
	Группа	Подпись, дата	И. О. Фамилия

Преподаватель	_____	Н. Б. Толпинская
---------------	-------	------------------

Преподаватель	_____	Ю. В. Строганов
	Подпись, дата	И. О. Фамилия

Москва, 2023 г.

1 Практические задания

1. Чем принципиально отличаются функции *cons*, *list*, *append*?

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

Каковы результаты вычисления следующих выражений?

```
1 (cons lst1 lst2)  
2 (list lst1 lst2)  
3 (append lst1 lst2)
```

- *cons* объединяет значения своих аргументов в точечную пару. Если вторым аргументом будет передан список, то в результате получится список, в котором второй аргумент будет добавлен в начало: ((A B)C D)
- *list* составляет из своих аргументов список: ((A B) (C D))
- *append* создает копию всех аргументов, кроме последнего, т. е. списковые ячейки. Связываются последними указателями. Результирующее значение: (A B C D)

2. Каковы результаты вычисления следующих выражений, и почему?

```
1 (reverse '(a b c))      —> Nil  
2 (reverse '(a b (c (d)))) —> Nil  
3 (reverse '(a))          —> (A)  
4 (last '(a b c))         —> (C)  
5 (last '(a))             —> (A)  
6 (last '((a b c)))       —> ((a b c))  
7 (reverse ())            —> Nil  
8 (reverse '((a b c)))    —> ((A B C))  
9 (last '(a b (c)))       —> ((c))  
10 (last ())              —> Nil
```

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

```
1 (defun get-last (lst)
2   (
3     car (last lst)
4   )
5 )
```

```
1 (defun get-last-reverse (lst)
2   (
3     car (reverse lst)
4   )
5 )
```

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

```
1 (defun get-last (lst)
2   (
3     nbutlast lst 1
4   )
5 )
```

```
1 (defun get-without-last-reverse (lst)
2   (
3     reverse (cdr (reverse lst))
4   )
5 )
```

5. Напишите функцию *swap-first-last*, которая переставляет в списке- аргументе первый и последний элементы.

```
1 (defun swap-first-last (lst)
2   (
3     nconc
4     (last lst)
5     (reverse
6       (cdr
7         (reverse (cdr lst)))
8     )
9     (list (car lst))
10  )
11 )
```

6. 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`.

```
1  (defvar first_player)
2  (defvar second_player)
3
4
5  (defun bones_throw ()
6
7    (print "Enter first bone: ")
8    (setq bone1 (read))
9    (print "Enter second bone: ")
10   (setq bone2 (read))
11   (setq ret (list bone1 bone2))
12   ret
13
14 )
15
16
17 (defun check-easy-win (lst)
18   (
19     or (= (+ (car lst) (cadr lst)) 7)
20        (= (+ (car lst) (cadr lst)) 11)
21   )
22 )
23
24 (defun pass_check (lst)
25   (
26     or
27     (
28       and (= (car lst) 1) (= (cadr lst) 1)
29     )
30     (
31       and (= (car lst) 6) (= (cadr lst) 6)
32     )
33   )
34 )
```

```

33     )
34 )
35
36
37 (defun play-game-second-player ()
38   (print "Second_player_throw:")
39   (setq second_player (bones_throw))
40   (print second_player)
41
42   (
43     cond
44     (
45       (check-easy-win second_player)
46       (print "Second_player_wins")
47     )
48     (
49       (pass_check second_player)
50       (play-game-second-player)
51     )
52     (
53       T
54       (
55         cond
56         (
57           (
58             >
59             (+ (car first_player) (cadr first_player))
60             (+ (car second_player) (cadr second_player))
61           )
62           (print "First_player_wins")
63         )
64         (
65           (
66             <
67             (+ (car first_player) (cadr first_player))
68             (+ (car second_player) (cadr second_player))
69           )
70           (print "Second_player_wins")
71         )
72         (
73           T

```

```

74         (print "Draw in the game")
75     )
76 )
77 )
78 )
79 )
80
81
82 (defun play-game-first-player ()
83
84     (print "First player throws:")
85     (setq first_player (bones_throw))
86     (print first_player)
87
88     (
89         cond
90         (
91             (check-easy-win first_player)
92             (print "First player wins")
93         )
94         (
95             (pass_check first_player)
96             (play-game-first-player)
97         )
98         (
99             T
100             (play-game-second-player)
101         )
102     )
103
104 )
105
106
107 (play-game-first-player)

```

7. Написать функцию, которая по своему списку-аргументу *lst* определяет является ли он палиндромом (то есть равны ли *lst* и *(reverse lst)*).

```

1 (defun get-without-last-reverse (lst)
2   (reverse (cdr (reverse lst))))
3 )
4

```

```

5
6
7  (defun st_check (lst)
8    (cond
9      (
10       (> (length lst) 1)
11       (and
12         (eq (car lst) (car (reverse lst)))
13         (st_check (cdr (get-without-last-reverse lst))))
14      )
15    )
16    (T T)
17  )
18 )
19
20
21 (defun palindrom_check (lst)
22   (st_check lst)
23 )

```

8. Напишите свои необходимые функции, которые обрабатывают таблицу из 4-х точечных пар: (страна . столица), и возвращают по стране - столицу, а по столице — страну.

```

1  (defun countries_capitals (lst name)
2    (
3      cond
4        (
5          (assoc name lst)
6          (cdr (assoc name lst))
7        )
8        (
9          (rassoc name lst)
10         (car (rassoc name lst))
11        )
12      (T Nil)
13    )
14  )

```

9. Напишите функцию, которая умножает на заданное число-аргумент первый числовой элемент списка из заданного 3-х элементного списка-

аргумента, когда а) все элементы списка — числа, б) элементы списка — любые объекты.

```
1  (defun mult_el_a (n lst)
2    (
3      cond
4      (
5        (and
6          (and
7            (numberp (car lst))
8            (and (numberp (cadr lst)) (numberp (caddr lst)))
9          )
10       (numberp n)
11     )
12     (* (car lst) n)
13   )
14   (T Nil)
15 )
16 )
```

```
1  (defun mult_el_b (n lst)
2    (cond
3      (
4        (and (numberp (car lst)) (numberp n))
5        (* (car lst) n)
6      )
7      (T Nil)
8    )
9  )
```