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?*

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

Каковы результаты вычисления следующих выражений?

```
(cons lstl lst2)
(list lst1 lst2)
(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 значе- ние: (A B C D)

2. *Каковы результаты вычисления следующих выражений, и почему?*

```
(reverse '(a b c))       --> Nil
(reverse '(a b (c (d)))) --> Nil
(reverse '(a))           --> (A)
(last '(a b c))      --> (C)
(last '(a))      --> (A)
(last '((a b c)))          --> ((a b c))
(reverse ()) --> Nil
(reverse '((a b c))) -->  ((A B C))
(last '(a b (c))) --> ((c))
(last ()) --> Nil
```

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

```
(defun get-last (lst)
  (
    car (last lst)
  )
)
```

```
(defun get-last-reverse (lst)
  (
    car (reverse lst)
  )
)
```

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

```
(defun get-last (lst)
  (
    nbutlast lst 1
  )
)
```

```
(defun get-without-last-reverse (lst)
  (
    reverse (cdr (reverse lst))
  )
)
```

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

```
(defun swap-first-last (lst)
  (
    nconc
    (last lst)
    (reverse
      (cdr
        (reverse (cdr lst))))
    )
    (list (car lst))
  )
)
```

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 большеочков. Результат игры и значения выпавших костей выводить на экран с помощью функции print.*

```
(defvar first_player)
(defvar second_player)


(defun bones_throw ()

  (print "Enter␣first␣bone:␣")
  (setq bone1 (read))
  (print "Enter␣second␣bone:␣")
  (setq bone2 (read))
  (setq ret (list bone1 bone2))
  ret

)


(defun check-easy-win (lst)
  (
    or ( = (+ (car lst) (cadr lst)) 7)
       ( = (+ (car lst) (cadr lst)) 11)
  )
)

(defun pass_check (lst)
  (
    or
    (
      and (= (car lst) 1) (= (cadr lst) 1)
    )
    (
      and (= (car lst) 6) (= (cadr lst) 6)
    )
```

```
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
```

```
                (print "Draw in the game")
              )
            )
          )
        )
    )


    (defun play-game-first-player ()

      (print "First player throws: ")
      (setq first_player (bones_throw))
      (print first_player)

      (
        cond
        (
          (check-easy-win first_player)
          (print "First player wins")
        )
        (
          (pass_check first_player)
          (play-game-first-player)
        )
        (
          T
          (play-game-second-player)
        )
      )

    )


    (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)).*

```
    (defun get-without-last-reverse (lst)
      (reverse (cdr (reverse lst)))
    )

```

```
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           (equal (caar lst) name)
6           (cdar lst)
7         )
8         (
9           (equal (cdar lst) name)
10          (caar lst)
11        )
12        (
13          T
14          (countries_capitals (cdr lst) name)
15        )
16      )
17    )
```

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.*

```
(defun mult_el_a (n lst)
  (
    cond
    (
      (
        and
        (
          and
          (numberp (car lst))
          (
            and
            (numberp (cadr lst))
            (numberp (caddr lst))
          )
        )
        (numberp n)
      )
      (* (car lst) n)
    )
    (
      T
      Nil
    )
  )
)
```

```
(defun mult_el_b (n lst)
  (
    cond
    (
      (
        and
        (numberp (car lst))
        (numberp n)
      )
      (* (car lst) n)
    )
```

```
12            (
13              T
14              Nil
15            )
16          )
17        )
```