



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»

на тему: «Использование управляющих структур, работа со списками»

Студент ИУ7-63Б
(Группа)

(Подпись, дата)

Лысцев Н. Д.
(И. О. Фамилия)

Преподаватель

(Подпись, дата)

Толпинская Н. Б.
(И. О. Фамилия)

Преподаватель

(Подпись, дата)

Строганов Ю. В.
(И. О. Фамилия)

2024 г.

1 Практические задания

1.1 Задание 1

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

Ответ:

- **cons** — двухаргументная базисная функция, которая создает отдельную списковую ячейку и распределяет указатели: **car**-указатель на первый аргумент, **cdr**-указатель — на второй элемент;
- **list** — не базисная функция (форма), принимающая произвольное количество аргументов и конструирующая список с переданными аргументами. Всегда возвращает список. Создается столько списковых ячеек, сколько аргументов;
- **append** — не базисная функция, не разрушающая структуру, которая принимает произвольное количество аргументов (списков) и объединяет их в один, при этом создавая копии всех переданных аргументов, кроме последнего, и возвращается новый список, содержащий комбинированные элементы.

Пусть `(setf lst1 '(a b)) (setf lst2 '(c d))`. Каковы результаты вычисления следующих выражений?

Листинг 1.1 – Решение задания №1

```
(cons lst1 lst2) ;; ((A B C) D E)
(list lst1 lst2) ;; ((A B C) (D E))
(append lst1 lst2) ;; (A B C D E)
```

1.2 Задание 2

Каковы результаты вычисления следующих выражений, и почему?

Листинг 1.2 – Решение задания №2 (начало)

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

Листинг 1.3 – Решение задания №2 (конец)

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

Ответ:

- **reverse** — функция, не разрушающая структуру. Возвращает новый список с элементами в обратном порядке. Работает только по верхнему уровню списковых ячеек;
- **last** — функция, принимающая один аргумент — список и возвращает последний элемент этого списка. Работает только по верхнему уровню списковых ячеек.

1.3 Задание 3

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

Листинг 1.4 – Решение задания №3

```
(defun get_last_elem(x)
  (car (last x)))

(defun get_last_elem2(x)
  (car (reverse x)))
```

1.4 Задание 4

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

Листинг 1.5 – Решение задания №4

```
(defun get_list_without_last(x)
  (reverse (cdr (reverse x))))

(defun get_list_without_last2(lst)
  (cond ((null (cdr lst)) nil)
        (t (cons (car lst) (get_list_without_last2 (cdr lst))))))
```

1.5 Задание 5

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

Листинг 1.6 – Решение задания №5

```
(defun swap-first-last(x)
  (let ((_first (car x))
        (_last (car (last x))))
    (setf (car x) _last)
    (setf (car (last x)) _first)))
```

1.6 Задание 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 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`.

Листинг 1.7 – Решение задания №6

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

(defun get_dice() ; функция для генерации значений игральной
  кости
  (list (+ (random 6) 1) (+ (random 6) 1)))

(defun is_repeat(dice)
  (let ((_first (car dice))
        (_second (car (cdr dice))))
    (cond ((or (= _first _second 1) (= _first _second 6)) t)
          (t nil))))

(defun player_actions(who)
  (let* ((_dice (get_dice))
         (_sum_dice (+ (car _dice) (car (cdr _dice)))))
    (print who)
    (print _dice)
    (cond ((or (= _sum_dice 7) (= _sum_dice 11)) 'win)
          ((is_repeat _dice) (player_actions))
          (t _sum_dice))))

(defun play_dice()
  (let ((_sum_dice1 (player_actions "first")))
    (cond ((eq _sum_dice1 'win) (print "first player is
      winner!"))
          (t (let ((_sum_dice2 (player_actions "second")))
                (cond ((eq _sum_dice2 'win) (print "second player is
                  winner!"))
                      ((< _sum_dice1 _sum_dice2) (print "second player is
                        winner!"))
                      ((> _sum_dice1 _sum_dice2) (print "first player is
                        winner!"))
                      (t (print "draw"))))))))))))
```

1.7 Задание 7

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

Листинг 1.8 – Решение задания №7

```
(defun list_is_pal(lst)
  (equal lst (reverse lst)))
```

1.8 Задание 8

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

Листинг 1.9 – Решение задания №8

```
(defun get_capital(table country)
  (cond ((null table) nil)
        ((eq (car (car table)) country) (cdr (car table)))
        (t (get_capital (cdr table) country))))

(defun get_country(table capital)
  (cond ((null table) nil)
        ((eq (cdr (car table)) capital) (car (car table)))
        (t (get_country (cdr table) capital))))
```

1.9 Задание 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.

Листинг 1.10 – Решение задания №9

```
(defun f(n lst) ; если все элементы списка -- числа
  (setf (car lst) (* (car lst) n)))

(defun f2(n lst) ; если все элементы списка -- любые объекты
  (cond ((null lst) nil)
        (t (cond ((numberp (car lst)) (setf (car lst) (* (car
    lst) n)))
              (t (f2 n (cdr lst)))))))
```