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 ИУ7-61Б (Группа) _____ (Подпись, дата) Постнов С. А. (Фамилия И. О.)

Преподаватель _____ (Подпись, дата) Толпинская Н. Б. (Фамилия И. О.)

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

2024 г.

СОДЕРЖАНИЕ

1 Практические задания

1.1 Задание 1

Ниже представлены отличия функций `cons`, `list` и `append`:

1) `cons` — базовая функция, которая объединяет значения двух своих аргументов в точечную пару;

2) `list` — принимает произвольное число аргументов и возвращает список, состоящий из значений аргументов.

3) `append` — не разрушающая структуру функция, которая объединяет списки и возвращает новый список, содержащий комбинированные элементы.

В листинге 1.1 приведены результаты вычисления выражений.

Листинг 1.1 – Результаты вычисления выражений

```
(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

В листинге 1.2 приведены результаты вычисления выражений.

Листинг 1.2 – Результаты вычисления выражений

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

1.3 Задание 3

В листинге 1.3 приведены два варианта функции, которая возвращает последний элемент своего списка-аргумента.

Листинг 1.3 – Функции, которые возвращают последний элемент своего списка-аргумента

```
(defun last_1 (l)
       (cond
            ((null l)
                        Nil)
            (t
                        (car (reverse l)))))

(defun last_2 (l)
       (cond
            ((cdr l)
                        (last_2 (cdr l)))
            (t
                        (cond
                             ((null l)
                                        Nil)
                             (t
                                        (car l))))))
```

1.4 Задание 4

В листинге 1.4 приведены два варианта функции, которая возвращает свой список аргумента без последнего элемента.

Листинг 1.4 – Функции, которые возвращают свой список аргумента без последнего элемента

```
(defun without_last_1 (l)
       (reverse
                (cdr
                        (reverse l))))

(defun without_last_2 (l)
       (cond
            ((null (cdr l))
                                Nil)
            (t
                                (cons (car l)
                                      (without_last_2 (cdr
                                        l))))))
```

1.5 Задание 5

В листинге 1.5 приведена функция `swap-first-last`, которая переставляет в списке-аргументе первый и последний элемент.

Листинг 1.5 – Функция `swap-first-last`

```
(defun without_last (l)
       (reverse
                (cdr
                        (reverse l))))

(defun swap-first-last (l)
    (cons (car (last l))
          (append (without_last (cdr l))
                  (cons (car l)
                        Nil))))
```

1.6 Задание 6

В листинге 1.6 приведена реализация простой игры в кости.

Листинг 1.6 – Реализация простой игры в кости

```
(defun get_sum (dice)
    (+ (car dice) (cdr dice)))

(defun roll_dice ()
    (setf *random-state* (make-random-state t))
    (cons (+ (random 5) 1)
          (+ (random 5) 1)))

(defun is_abs_win (dice)
    (let ((sum (get_sum dice)))
         (or (= sum 7)
             (= sum 11))))

(defun is_reroll (dice)
    (let ((fir (car dice))
          (sec (cdr dice)))
         (or (and (= fir 6) (= sec 6))
             (and (= fir 1) (= sec 1)))))


```

```
(defun get_dice (id)
    (let ((dices (roll_dice)))
          (princ id)
          (princ dices)
         (cond
              ((is_reroll dices)
                                  (get_dice id))
               (t
                                  dices))))

(defun play ()
    (let ((dice1 (get_dice 1))
          (dice2 (get_dice 2)))
         (cond
              ((is_abs_win dice1)
                                                (print
                                                  "1_win_abs"))
              ((is_abs_win dice2)
                                                (print
                                                  "2_win_abs"))
              ((> (get_sum dice1) (get_sum dice2))
                                                (print "1_win"))
              ((< (get_sum dice1) (get_sum dice2))
                                                (print "2_win"))
              (t
                                                (print
                                                  "draw")))))
```

1.7 Задание 7

В листинге 1.7 приведена функция, которая по своему списку-аргументу `lst` определяет, является ли он палиндромом.

Листинг 1.7 – Функция, которая определяет, полиндром ли список

```
(defun palindrom (lst)
    (equal lst
           (reverse lst)))
```

1.8 Задание 8

В листинге 1.8 приведены функции, которые обрабатывают таблицу из 4-х точечных пар: (страна . столица), и возвращают по стране — столицу, а по столице — страну.

Листинг 1.8 – Функции, которые обрабатывают таблицу из 4-х точечных пар

```
(set 'table (list
                  (cons 'Russia 'Moscow)
                  (cons 'Georgia 'Tbilisi)
                  (cons 'Japan 'Tokio)
                  (cons 'China 'Pekin)))

(defun get_capital (table country)
       (cond
            ((null table)
                                        Nil)
            ((eql country (caar table))
                                        (cdar table))
            (t
                                        (get_capital (cdr table)
                                           country))))

(defun get_country (table capital)
       (cond
            ((null table)
                                        Nil)
            ((eql capital (cdar table))
                                        (caar table))
            (t
                                        (get_country (cdr table)
                                           capital))))
```

1.9 Задание 9

В листинге 1.9 приведена функция, которая умножает на заданное число-аргумент первый числовой элемент списка из заданного 3-х элементного списка-аргумента, когда:

1) все элементы списка — числа;

2) элементы списка — любые объекты.

Листинг 1.9 – Функция, которая умножает на заданное число-аргумент первый числовой элемент списка

```
(defun multi (lst x)
       (cond
            ((null lst)
                                  Nil)
            ((numberp (car lst))
                                  (cons (* (car lst) x)
                                        (cdr lst)))
            (t
                                  (cons (car lst)
                                        (multi (cdr lst) x)))))
```