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 № 5

по курсу «Функциональное и логическое программирование»

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

Студент ИУ7-61Б _____ Мицевич М. Д.
(Группа) (Подпись, дата) (И. О. Фамилия)

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

2022 г.

Задание 1

Постановка задачи

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

Решение

Листинг 1 – Решение задания №1

```
(defun list-compare (lst1 lst2)
    (cond ((and (null lst1) (null lst2)) T)
          ((eql (car lst1) (car lst2)) (list-compare (cdr lst1) (
             cdr lst2)))))

(defun is-polyndrome (lst) (list-compare lst (reverse lst)))
```

Задание №2

Постановка задачи

Написать предикат `set-equal`, который возвращает `t`, если два его множества-аргумента содержат одни и те же элементы, порядок которых не имеет значения

Решение

Листинг 2 – Решение задания №2

```
(defun is-included (inlst outlst)
    (cond ((null inlst) T)
          ((member (car inlst) outlst) (is-included (cdr inlst)
             outlst))))

(defun set-equal (lst1 lst2)
    (and (eq (length lst1) (length lst2))
         (is-included lst1 lst2)
         (is-included lst2 lst1)))
```

Задание №3

Постановка задачи

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

Решение

Листинг 3 – Решение задания №3

```
(defun get-capital (table country)
    (cond ((null table) nil)
          ((equal (caar table) country) (cdar table))
          (T (get-capital (cdr table) country))))

(defun get-country (table capital)
    (cond ((null table) nil)
          ((equal (cdar table) capital) (caar table))
          (T (get-country (cdr table) capital))))
```

Задание №4

Постановка задачи

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

Решение

Листинг 4 – Решение задания №4

```
(defun swap-first-last (lst)
    (append (last lst)
            (cdr (reverse (cdr (reverse lst))))
            (cons (car lst) nil)))
```

Задание №5

Постановка задачи

Напишите функцию `swap-two-ellement`, которая переставляет в списке-аргументе два указанных своими порядковыми номерами элемента в этом списке

Решение

Листинг 5 – Решение задания №5

```
(defun append-element (lst el)
    (append lst (cons el nil)))

(defun swap (lst el1 el2 pos1 pos2 res ind)
    (cond ((null lst) res)
          ((eql pos1 ind) (swap (cdr lst) el1 el2 pos1 pos2 (
             append-element res el2) (+ ind 1)))
          ((eql pos2 ind) (swap (cdr lst) el1 el2 pos1 pos2 (
             append-element res el1) (+ ind 1)))
          (T (swap (cdr lst) el1 el2 pos1 pos2 (append-element
             res (car lst)) (+ ind 1)))))

(defun swap-two-elements (lst pos1 pos2)
    (swap lst (nth pos1 lst) (nth pos2 lst) pos1 pos2 () 0))
```

Задание №6

Постановка задачи

Напишите две функции, `swap-to-left` и `swap-to-right`, которые производят круговую перестановку в списке-аргументе влево и вправо, соответственно

Решение

Листинг 6 – Решение задания №6

```
(defun append-element (lst el)
    (append lst (cons el nil)))
```

```
(defun swap-to-left (lst)
    (append-element (cdr lst) (car lst)))

(defun swap-to-right (lst)
    (append (last lst) (reverse (cdr (reverse lst)))))
```

Задание №7

Постановка задачи

Напишите функцию, которая добавляет к множеству двухэлементных списков новый двухэлементный список, если его там нет.

Решение

Листинг 7 – Решение задания №7

```
(defun append-element (lst el)
    (append lst (cons el nil)))

(defun existp (lst pair)
    (cond ((null lst) nil)
          ((and (eql (caar lst) (car pair)) (equal (cdar lst) (
             cdr pair))) T)
          (T (existp (cdr lst) pair))))

(defun add-to-set (pair lst)
    (if (existp lst pair)
        lst
        (append-element lst pair)))
```

Задание №8

Постановка задачи

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

Решение

Листинг 8 – Решение задания №8

```
(defun mult-only-numbers (lst num)
    (cons (* (car lst) num) (cdr lst)))

(defun mult-first-number (lst num)
    (cond ((numberp (first lst)) (mult-only-numbers lst num))
          ((numberp (second lst)) (list (first lst) (* (second
             lst) num) (third lst)))
          ((numberp (third lst)) (list (first lst) (second lst)
             (* (third lst) num)))))
```

Задание №9

Постановка задачи

Напишите функцию, select-between, которая из списка-аргумента из 5 чисел выбирает только те, которые расположены между двумя указанными границами-аргументами и возвращает их в виде списка.

Решение

Листинг 9 – Решение задания №9

```
(defun betweenp (el b1 b2)
    (< (* (- el b1) (- el b2)) 0))

(defun select-between (lst b1 b2)
    (cond ((null lst) nil)
          ((betweenp (car lst) b1 b2) (cons (car lst) (
             select-between (cdr lst) b1 b2)))
          (T (select-between (cdr lst) b1 b2))))
```

Контрольные вопросы

Вопрос 1. Структуроразрушающие и не разрушающие структуру списка функции.

Ответ. Функции, реализующие опреации со списками, делятся на две группы:

1. не разрушающие структуру функции; данные функции не меняют переданный им объект-аргумент, а создают копию, с которой в дальнейшем производят необходимые преобразования; к таким функциям относятся: append, reverse, last, nth, nthcdr, length, remove, subst и др.

2. структуроразрушающие функции; данные функции меняют сам объект-аргумент, из-за чего теряется возможность работать с исходным списком; чаще всего имя структуроразрушающих функций начинается с префикса -n: nreverse, nconc, nsubst и др.

Обычно в Lisp существуют функции-дубли, которые реализуют одно и то же преобразование, но по разному (с сохранением структуры и без): apppend/nconc, reverse/nreverse и т.д.

Вопрос 2. Отличие в работе функций cons, list, append, nconc и в их результате.

Ответ. Функция **cons** - чисто математическая, она принимает ровно 2 аргумента, создает бианрный узел и расставляет указатели (car - на первый аргумент, cdr - на второй). В результате работы функции может получится как точечная пара, так и список (зависит от второго аргумента).

Функция **list** - это форма, она принимает произвольное количество аргументов и создает из них список. В отличии от функции cons, list создает столько бинарных узлов, сколько передано ей аргументов, и связывает их вместе. Результатом работы данной функции всегда будет список.

Функция **append** также является формой. Она принимает на вход произвольное число аргументов. Для всех аргументов, кроме последнего, эта функция создает копию, ссылая при этом последний элемент каждого списка

аргумента на первый элемент следующего по порядку списка аргумента. В результате работы функции append может получится как список, так и точечная пара (зависит от последнего аргумента).

Итого: **cons** создает однин бинарный узел, **list** создает столько бинарных узлов, сколько передано аргументов, **append** создает копии всех бинарных узлов для каждого из аргументов, исключая последний аргумент.