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»

Отчёт по лабораторной работе №5 по курсу «Функциональное и логическое программирование»

Тема Использование управляющих структур, работа со списками

Студент Прянишников А.Н.

Группа ИУ7-65Б

Оценка (баллы) _____

Преподаватели Строганов Ю. В., Толпинская Н. Б.

Москва — 2022 г.

Практическое задание

Задание 1

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

```
(defun isPalindrom (lst)
(if (= (length lst) 1) T
(if (= (length lst) 2) (= (second lst) (first lst))
    (and (= (car (last lst)) (car lst)) (isPalindrom (butlast (cdr lst)))))))
```

Задание 2

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

```
(defun set-equal (lst1 lst2)
(and (subsetp lst1 lst2) (subsetp lst2 lst1)))
```

Задание 3

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

```
(defun find-country (map findedCity)
(if (equal (car map) Nil) Nil
(if (equal (cadar map) findedCity)
    (caar map)
    (find-capital (cdr map) findedCity))))

(defun find-capital (map findedCountry)
(if (equal (car map) Nil) Nil
(if (equal (caar map) findedCountry)
    (cadar map) (find-capital (cdr map) findedCountry))))
```

Задание 4

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

```
(defun swap-first-last (lst)
(setf (cdr (last lst)) (car lst))
(cdr lst))
```

Задание 5

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

```
(defun swap-two-elements (i j lst)
  (let ((lst-copy (copy-list lst)))
    (and (< i (length lst)) (< j (length lst))
         (let ((el1 (nth i lst))
               (el2 (nth j lst)))
           (setf (nth i lst-copy) el2)
           (setf (nth j lst-copy) el1)
           lst-copy))))
```

Задание 6

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

```
(defun swap-to-left (lst)
(reverse (cons (car lst) (reverse (cdr lst)))))

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

Задание 7

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

```
(defun check (lst1 addlst)
(if (equal (cdr lst1) Nil) T
(if (equal (car lst1) addlst) Nil (add-list (cdr lst1) addlst))))

(defun add-list (lst1 addlst)
(if (check lst1 addlst) (and (setf (cdr (last lst1)) addlst) lst1) lst1))
```

Задание 8

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

- все элементы списка — числа;
- элементы списка – любые объекты.

```
(defun multiply-first (array koef)
(and (setf (car array) (* (car array) koef)) array))

(defun multiply-first (array koef)
(if (numberp (first array)) (and (setf (car array) (* (car array) koef) )array)
(if (numberp (second array)) (and (setf (cadr array) (* (cadr array) koef) ) array)
(if (numberp (third array)) (and (setf (caddr array) (* (caddr array) koef)) array)))))
```

Задание 9

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 (упорядоченного по возрастанию списка чисел (+ 2 балла)).

```
(defun select-between (a b lst)
(sort
(mapcan #'(lambda (x) (and (>= x a) (<= x b) (list x))) lst)
#'<))
```

Теоретические вопросы

1. Структуроразрушающие и не разрушающие структуру списка функции

Не разрушающие структуру списка функции

- `append` — Объединяет списки. Создает копию для всех аргументов, кроме последнего;
- `reverse` — Возвращает копию исходного списка, элементы которого переставлены в обратном порядке (работает только на верхнем уровне);
- `last` — Возвращает последнюю списковую ячейку верхнего уровня;
- `nth` — Возвращает указателя от n-ной списковой ячейки. Нумерация начинается с нуля;
- `nthcdr` — Возвращает n-ого хвоста;
- `length` — Возвращает длину списка (верхнего уровня);
- `remove` — Данная функция удаляет элемент по значению (работает с копией), можно передать функцию сравнения через `:test`;
- `subst` — Заменяет все элементы списка, которые равны 2 переданному элементу-аргументу на другой 1 элемент-аргумент.

Структуроразрушающие функции

Данные функции меняют сам объект-аргумент, невозможно вернуться к исходному списку. Чаще всего такие функции начинаются с префикса `n-`.

- `nconc` — Работает аналогично `append`, только не копирует свои аргументы, а разрушает структуру;

- `nreverse` — Работает аналогично `reverse`, но не создает копии;
- `nsubst` — Работает аналогично функции `nsubst`, но не создает копии;

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

Функция `cons` cons создает списковую ячейку и ставит указатели на 2 аргумента, таким образом объединяя свои аргументы в точечную пару.

Примеры:

1. `(cons 1 '(2 3))` — `(1 2 3)`;
2. `(cons '(2 3) 1)` — `((2 3) . 1)`.

Функция `list` принимает произвольное число аргументов. Функция возвращает список, состоящий из значений аргументов.

`(list 1 2 3)` — `(1 2 3)`;

`(list 2 '(1 2))` — `(2 (1 2))`;

`(list '(1 2) '(3 4))` — `((1 2) (3 4))`;

Функция `append` — форма, принимает на вход произвольное количество аргументов и для всех аргументов, кроме последнего, создает копию, ссылая при этом последний элемент каждого списка-аргумента на первый элемент следующего по порядку списка-аргумента.

`(append '(1 2) '(3 4))` — `(1 2 3 4)`;

`(append '((1 2) (3 4)) '(5 6))` — `((1 2) (3 4) 5 6)`.

Функция `nconc` принимает на вход произвольное число аргументов, каждый из которых должен быть списком, кроме последнего. Функция возвращает список, который образован конкатенацией всех аргументов. Результат сохраняется не в копию массива, а в первый аргумент.

`(nconc '(1 2) '(3 4) '(5 6))` — `(1 2 3 4 5 6)`;

`(nconc '((1 2) (3 4)) '(5 6))` — `((1 2) (3 4) 5 6)`.