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"

Тема Использование управляющих структур, работа со списками

Студент Варин Д.В.

Группа ИУ7-66Б

Оценка (баллы) _____

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

Москва — 2022 г.

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

Листинг 1 – Задание 1

```
(defun palindromp (lst) (equal lst (reverse lst)))

(palindromp '(1 2))
(palindromp '(1 2 1))
```

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

Листинг 2 – Задание 2

```
(defun set-equal (l1 l2)
  (and (subsetp l1 l2) (subsetp l2 l1)))

(set-equal '(1 2 3) '(3 1 2))
(set-equal '(1 2 3) '(1 2 3 4))
```

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

Листинг 3 – Задание 3

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

```
  (T ())))

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

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

Листинг 4 – Задание 4

```
(defun swap-first-last (lst)
  (if lst
      (let ((first (car lst))
      (last (car (last lst))))
  (setf (car lst) last)
  (setf (car (last lst)) first)
  lst)))

(swap-first-last '(1 5 5 5 3))
(swap-first-last '(1))
(swap-first-last ())
```

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

Листинг 5 – Задание 5

```
(defun swap-two-element (lst n1 n2)
  (if (and lst (< n1 (length lst)) (< n2 (length lst)))
      (let ((n1th (nth n1 lst))
      (n2th (nth n2 lst)))
  (setf (nth n1 lst) n2th)
  (setf (nth n2 lst) n1th)
```

```
  lst)))

(swap-two-element '(0 1 2 3 4 5) 1 3)
(swap-two-element () 0 0 )
(swap-two-element '(0 1 2) 1 3)
```

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

Листинг 6 – Задание 6

```
(defun swap-to-left (lst)
  (if lst
      (progn
  (setf (cdr (last lst)) (cons (car lst) ()))
  (cdr lst))))

;
(defun swap-to-left (lst)
  (let ((first (car lst)))
    (reverse (cons first (reverse (cdr lst))))))

(swap-to-left '(0 1 2 3))
(swap-to-left '(1))
(swap-to-left ())

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


(swap-to-right '(0 1 2 3))
(swap-to-right '(1))
(swap-to-right ())
```

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

Листинг 7 – Задание 7

```
(defun add-2list (src lst)
  (cond ((not src) (setf src (cons lst ())))
  ((not (cdr src)) (setf (cdr src) (cons lst ())))
  ((equal (car src) lst))
  (T (add-2list (cdr src) lst)))
  src)

(add-2list '((1 2) (3 4) (5 6)) '(1 2))
(add-2list '() '(1 2))
```

8. 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 8 – Задание 8

```
(defun mult-first (lst n)
  (cons (* (car lst) n) (cdr lst)))

(mult-first '(3 2 1) 3)

(defun mult-first-all (lst n)
  (cond ((not lst) ())
  ((numberp (car lst)) (setf (car lst) (* (car lst) n)))
  (T (mult-first-all (cdr lst) n)))
  lst)

(mult-first-all '(3 2 1) 3)
(mult-first-all '(a 2 1) 3)
```

```
(mult-first-all '(a b) 3)
(mult-first-all () 3)
```

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

Листинг 9 – Задание 9

```
(defun select-between (lst n1 n2)
  (cond ((> n1 n2) (select-between lst n2 n1))
  ((or (>= n1 (length lst)) (>= n2 (length lst))) ())
  (T (reverse (nthcdr (- (length lst) n2 1) (reverse (nthcdr n1 lst))))
   )))

(select-between '(0 1 2 3 4) 0 3)
(select-between '(0 1 2 3 4) 3 0)
(select-between '(0 1 2 3 4) 3 3)
(select-between () 3 3)
(select-between '(0 1 2) 3 4)

(defun select-between-sorted (lst n1 n2)
  (sort (select-between lst n1 n2) #'<))

(select-between-sorted '(0 3 1 2 4) 0 3)
```

Контрольные 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 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`;
- `rplaca` — Переставляет `car`-указатель на 2 элемент-аргумент (S-выражение);
- `rplacd` — Переставляет `cdr`-указатель на 2 элемент-аргумент (S-выражение);
- `subst` — Заменяет все элементы списка, которые равны 2 переданному элементу-аргументу на другой 1 элемент-аргумент. *По умолчанию для сравнения используется функция `eql`.*

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` — Работае аналогично функции `nsubst`, но не создает копии;

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

Функция `cons` — чисто математическая, конструирует списковую ячейку, которая может вовсе и не быть списком (будет списком только в том случае, если 2 аргументом передан список).

Примеры:

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

Функция `list` — форма, принимает произвольное количество аргументов и конструирует из них список. Результат — всегда список. При нуле аргументов возвращает пустой список.

`(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 (так как модифицируются все списки-аргументы, кроме последнего, копирование для последнего не делается в целях эффективности).

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

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