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 Богаченко А. Е.

Группа ИУ7-65Б

Оценка (баллы) _____

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

Москва — 2022 г.

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

Листинг 1 – Задание 1

```
(defun palindromep (lst)
  (equal lst (reverse lst)))
```

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 (lst1 lst2)
  (and (subsetp lst2 lst1) (subsetp lst1 lst2)))
```

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 ->capital (country country-capital)
  (let ((pair (assoc country country-capital)))
    (and pair (cdr pair))))

(defun ->country (capital country-capital)
  (let ((pair (rassoc capital country-capital)))
    (and pair (car pair))))
```

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

Листинг 4 – Задание 4

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

(defun swap-first-last (lst)
  (let ((el (car lst))
        (last (car (last lst))))
    (reverse (cons el (cdr (reverse (cons last (cdr lst))))))))
```

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! (n1 n2 lst)
  (let ((len (length lst)))
    (and (< n1 len) (< n2 len)
         (let ((el1 (nth n1 lst))
               (el2 (nth n2 lst)))
           (setf (nth n1 lst) el2)
           (setf (nth n2 lst) el1)
           lst))))

(defun swap-two (n1 n2 lst)
  (let ((len (length lst))
        (lst-copy (copy-list lst)))
    (and (< n1 len) (< n2 len)
         (let ((el1 (nth n1 lst))
               (el2 (nth n2 lst)))
           (setf (nth n1 lst-copy) el2)
           (setf (nth n2 lst-copy) el1)
           lst-copy))))
```

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

Листинг 6 – Задание 6

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

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

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)`.