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

к лабораторной работе №7

По курсу: «Функциональное и логическое программирование»

Тема: «Использование управляющих структур, модификация списков».

Студент: Якуба Д.В.
Группа: ИУ7-63Б
Преподаватели: Толпинская Н. Б.,
Строганов Ю. В.

Москва, 2021 г.

Практическая часть

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

Решение:

```
(defun inner-reverse (lst-in lst-out)
    (cond
        ((null lst-in) lst-out)
        (t (inner-reverse (cdr lst-in) (cons (car lst-in) lst-out)))))

(defun my-reverse (lst-in)
    (inner-reverse (cdr lst-in) (cons (car lst-in) nil)))

(defun isPal (lst)
    (equal lst (my-reverse lst)))
```

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

Решение:

```
(defun is-in-set (lst el)
    (cond
        ((null lst) nil)
        ((equal (car lst) el))
        (t (is-in-set (cdr lst) el))))

(defun set-equal-inner (stF stS)
    (cond
        ((null (cdr stF)) (is-in-set stS (car stF)))
        ((is-in-set stS (car stF)) (set-equal-inner (cdr stF) stS))))

; Функция, пародирующая поведение length
(defun len-inner (lst acc)
    (cond
        ((null lst) acc)
        (t (len-inner (cdr lst) (+ acc 1)))))

(defun len (lst)
    (len-inner lst 0))
;//

(defun set-equal (stF stS)
    (cond
        ((= (len stF) (len stS)) (set-equal-inner stF stS))))
```

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

Решение:

```
(defun find-country (table capital)
    (cond ((null table) '(Такой столицы нет в таблице))
          ((eq (cdar table) capital) (caar table))
          (t (find-country (cdr table) capital))))

(defun find-capital (table country)
    (cond ((null table) '(Такой страны нет в таблице))
          ((eq (caar table) country) (cdar table))
          (t (find-capital (cdr table) country))))
```

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

Решение:

Представлено две реализации: с cons-дополняемой рекурсией и хвостовой рекурсией.

```
(defun my-last (lst)
    (cond
        ((null (cdr lst)) lst)
        (t (my-last (cdr lst)))))

(defun swap-first-last-inner (lst first-el)
    (cond
        ((null (cdr lst)) (cons first-el nil))
        (t (cons (car lst) (swap-first-last-inner (cdr lst) first-el)))))

(defun swap-first-last (lst)
    (cons (car (my-last lst)) (swap-first-last-inner (cdr lst) (car lst))))
```

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

```
(defun my-nth (lst index)
    (cond
        ((= 0 index) (car lst))
        (t (my-nth (cdr lst) (- index 1)))))

(defun len-inner (lst acc)
    (cond
        ((null lst) acc)
        (t (len-inner (cdr lst) (+ acc 1)))))

(defun len (lst)
    (len-inner lst 0))

; Функция прохода до конца списка
(defun swap-two-elements-nil (lst)
    (cond
        ((null lst) nil)
        (t (cons (car lst)
                        (swap-two-elements-nil (cdr lst))))))


(см. сл. стр.)
```

```
; Функция прохода до второго индекса
(defun swap-two-elements-last (lst indS f-el)
    (cond
        ((= 0 indS) (cons f-el
                          (swap-two-elements-nil (cdr lst))))
        (t (cons (car lst)
                 (swap-two-elements-last (cdr lst) (- indS 1) f-el)))))

; Функция прохода до первого индекса
(defun swap-two-elements-inner (lst indF indS s-el)
    (cond
        ((= 0 indF) (cons s-el
                          (swap-two-elements-last (cdr lst) indS (car lst))))
        (t (cons (car lst)
                 (swap-two-elements-inner (cdr lst) (- indF 1) indS s-
el)))))

; Обёрточная функция
(defun swap-two-elements (lst indF indS)
    (and
        (and (>= indF 0) (>= indS 0) (< indF (len lst)) (< indS (len lst)))
        (cond
            ((< indF indS)
                (swap-two-elements-inner lst indF (- indS indF 1) (my-
nth lst indS)))
            (t (swap-two-elements-inner lst indS (- indF indS 1) (my-
nth lst indF))))))
```

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

Решение:

```
(defun my-last (lst)
    (cond
        ((null (cdr lst)) lst)
        (t (my-last (cdr lst)))))

(defun swap-to-left-inner (lst f-el)
    (cond
        ((null lst) (cons f-el nil))
        (t (cons (car lst) (swap-to-left-inner (cdr lst) f-el)))))

(defun swap-to-left (lst)
    (swap-to-left-inner (cdr lst) (car lst)))

(defun swap-to-right-inner (lst)
    (cond
        ((null (cdr lst)) nil)
        (t (cons (car lst) (swap-to-right-inner (cdr lst))))))

(defun swap-to-right (lst)
    (cons (my-last lst) (swap-to-right-inner lst)))
```

Теоретическая часть

1. Способы определения функций.

Первый способ:

```
(defun *имя функции* (*список параметров*) (
      *тело функции*
      )
)
```

Пример:

```
(defun findCat (gip cat)
    (sqrt ( - (* gip gip) (* cat cat))))

(findCat 5 4) -> 3.0;
```

Второй способ:

```
(lambda (*список аргументов*)) (*тело функции*))
```

Пример:

```
((lambda (a) (* a 3)) 4) -> 12
```

lamda-функции называются «безымянными». Суть такой функции состоит в том, что задается алгоритм вычисления, но не задается имени функции. Подобную функцию можно применить к списку аргументов и сразу получить результат.

2. Варианты и методы модификации элементов списка.

Существует два вида функций работы со списками: структуроразрушающие и не разрушающие структуру функции.

Структуроразрушающими называются функции, при помощи которых можно вносить изменения во внутреннюю структуру уже существующих выражений.

К структуроразрушающим функциям относят: nreverse, rplaca, rplacd, nconc, nsubst и т.д.

К не разрушающим структуру функциям относят: append, reverse, remove и т.д.