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 (ИУ7)**

ОТЧЕТ

по лабораторной работе № 7

Название: Использование управляющих структур, модификация списков.

Дисциплина: Функциональное и логическое программирование

Студент ИУ7-63Б _____ В.П. Федоров
(Группа) (Подпись, дата) (И.О. Фамилия)

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

Москва, 2021

Краткие теоретические сведения.

Многие стандартные функции Lisp являются формами и реализуют особый способ работы со своими аргументами. К таким функциям относятся функции, позволяющие работать с произвольным количеством аргументов: and, or, append, или
особым образом обрабатывающие свои аргументы: cond, if, append, remove, reverse и др.

Если на вход функции подается структура данных (список), то возникает вопрос: сохранится ли возможность в дальнейшем работать с исходными структурами, или они изменятся в процессе реализации функции. В Lisp существуют функции, использующие списки в качестве аргументов и разрушающие или не разрушающие структуру, исходных аргументов при этом часть из них позволяет использовать произвольное количество аргументов, а часть нет.

Цель работы:

Сравнить особенности работы функций CONS, LIST, APPEND, NCON на аргументах разной структуры.

Задание №1.
Написать функцию, которая по своему списку-аргументу lst определяет является ли он палиндромом (то есть равны ли lst и (reverse lst)).

```
(defun is-lst-palindrom(lst)
        (if (equal lst (reverse lst)) t))
```

```
* (IS-LST-PALINDROM '(1 2 3))
NIL
* (IS-LST-PALINDROM '(1 2 1))
T
```

Задание №2.
Написать функцию-предикат set-equal, который возвращает t, если два его множества-аргумента содержат одни и те же элементы, порядок которых не имеет значения.

```
(defun set-equal(setA setB)
        (if (equal (sort setA #'<) (sort setB #'<)) t))
```

```
* (set-equal '(3 2 1 4) '(1 2 3 4))
T
* (set-equal '(1 2 3) '(0 0 0))
```

NIL

Задание №3.
Написать необходимые функции, которые обрабатывают таблицу из точечных пар: (страна.столица), и возвращают по стране - столицу, а по столице - страну.

```
(defun get-country(table capital)
        (cond
                ((null table) 'no-info)
                ((eq (cdr (car table)) capital) (car (car table)))
                (t (get-country (cdr table) capital))))
```

* (country '((Russia . Moscow) (USA . Washington) (Spain . Madrid) (Italy . Rome)) 'Moscow)
RUSSIA

* (country '((Russia . Moscow) (USA . Washington) (Spain . Madrid) (Italy . Rome)) 'Rome)
ITALY

```
(defun get-capital(table country)
        (cond
                ((null table) 'no-info)
                ((eq (car (car table)) country) (cdr (car table)))
                (t (get-capital (cdr table) country))))
```

* (get-capital '((Russia . Moscow) (USA . Washington) (Spain . Madrid) (Italy . Rome)) 'Russia)
MOSCOW
* (get-capital '((Russia . Moscow) (USA . Washington) (Spain . Madrid) (Italy . Rome)) 'USA)
WASHINGTON
* (get-capital '((Russia . Moscow) (USA . Washington) (Spain . Madrid) (Italy . Rome)) 'Brazil)
NO-INFO

Задание №4.
Написать функцию swap-first-last, которая переставляет в списке-аргументе первый и последний элементы.

```
(defun delete-last(lst)
        (reverse (cdr (reverse lst))))

(defun delete-first(lst)
        (cdr lst))

(defun get-center(lst)
        (delete-first (delete-last lst)))

(defun swap-first-last(lst)
        (append (last lst) (get-center lst) (list (car lst))))
```

```
* (swap-first-last '(1 2 3 4 5))
(5 2 3 4 1)
```

Задание №5.
Написать функцию swap-two-element, которая переставляет в списке-аргументе два указанных своими порядковыми номерами элемента в этом списке.

```
(defun swipe-two-elements(lst a b)
        (append (subseq lst 0 a) (list (nth b lst)) (subseq lst (+ a 1) b)
                                        (list (nth a lst)) (subseq lst (+ b 1))))
```

Задание №6.
Написать 2 функции, swap-to-left и swap-to-right, которые производят круговую перестановку в списке-аргументе влево и вправо соответственно.

Листинг 6: Циклический сдвиг влево

```
(defun swipe-left(lst)
        (append (cdr lst) (list (car lst))))

(defun circle-swipe-left(lst k)
        (cond
                ((= k 0) lst)
                (t (circle-swipe-left (swipe-left lst) (- k 1)))))
```

Листинг 7: Циклический сдвиг вправо

```
(defun swipe-right(lst)
        (cons (nth (- (length lst) 1) lst) (subseq lst 0 (- (length lst) 1))))

(defun circle-swipe-right(lst k)
```

```
(cond
        ((= k 0) lst)
        (t (circle-swipe-right (swipe-right lst) (- k 1)))))
```

Теоретические вопросы.

1. *Способы определения функций.*

Обычно функции определяются с помощью макроса defun.
Листинг 4.1: типовое использование макроса defun.

```
(defun name (parameter*)
    тело-функции*)
```

В качестве имени можно использовать любой символ, но обычно используются только буквы, цифры и знак минус. Рекомендуются избегать символа нижнего подчеркивания.

Список параметров функции определяет переменные, которые будут использоваться для хранения аргументов, переданных при вызове функции.

Тело defun состоит из произвольного числа s-выражений. При выполнении функции они будут выполнены по порядку, и будет возвращен результат последнего (в качестве результата работа всей функции).

Возможны ситуации, в которых определению новых функций при помощь defun является излишним. Для таких ситуаций в Lisp предусмотрена возможность создания анонимных функций при помощи выражения lambda.

Листинг 4.2: создание анонимной функции.

```
(lambda (parameters) body)
```