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 дисциплине «Функциональное и логическое программирование»

Тема Использование управляющих структур, работа со списками

Студент Варламова Е. А.

Группа ИУ7-61Б

Оценка (баллы) _____

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

Москва — 2022 г.

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

Листинг 1: Решение задания №1

```
(defun move (lst res)
(cond ((null lst) res)
      (t (move (cdr lst) (cons (car lst) res) ))
) )

(defun my_reverse (lst)
(move lst ()))

(defun is_eq (lst revlst)
(cond ((and (null lst) (null revlst)))
      ( (eq (car lst) (car revlst)) (is_eq (cdr lst) (cdr revlst)) )
))

(defun is_poly (lst)
(is_eq lst (my_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: Решение задания №2

```
(defun find_element (el lst)
(cond
((null lst) NIL)
((eql el (car lst)) T)
(T (find_element el (cdr lst)))))

(defun compare (lst1 lst2)
(cond
((null lst1) T)
((find_element (car lst1) lst2) (compare (cdr lst1) lst2))
(T NIL)))

(defun are_equal (lst1 lst2) (if (=(length lst1) (length lst2)) (compare
   lst1 lst2)))
```

3. Напишите свои необходимые функции, которые обрабатывают таблицу из 4-х точечных пар:

(страна . столица), и возвращают по стране - столицу, а по столице — страну .

Листинг 3: Решение задания №3

```
(defun get_capital (lst country)
(reduce #'(lambda (x y)
(if (null x)
```

```
(if (eql (car y) country) (cdr y))
 x)) lst :initial-value Nil))

(defun get_country (lst capital)
(reduce #'(lambda (x y)
(if (null x)
(if (eql (cdr y) capital) (car y))
 x)) lst :initial-value Nil))
```

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

Листинг 4: Решение задания №4

```
(defun swap-first-last (lst)
(setf f (car lst))
(setf nl (get_change_last (cdr lst)))
(cons l nl)
)

(defun get_change_last (lst)
(if (cdr lst)
(cons (car lst) (get_change_last (cdr lst)))
((lambda () (setf l (car lst)) (list f) ))
))
```

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

Листинг 5: Решение задания №5

```
(defun without_last_els (lst pos ind)
(cond
( (< ind pos) (cons (car lst) (without_last_els (cdr lst) pos (+ ind 1)) ) )
( (= ind pos) (lambda ()
(setf tail (cdr lst))
(cons (car lst) (without_last_els (cdr lst) pos (+ ind 1) ))
))
))

(defun f (lst p1 p2 ind)
(cond
( (< ind (- p1 1)) (cons (car lst) (f (cdr lst) p1 p2 (+ ind 1))))
( (= ind (- p1 1)) (cons (car lst) (append (swap-first-last (
   without_last_els (cdr lst) p2 p1)) tail) ) )
))

(defun swap-two-element (lst p1 p2)
(cond ((< p1 p2) (f lst p1 p2 1))
( (> p1 p2) (f lst p2 p1 1))
```

```
(T lst)
))
```

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

Листинг 6: Решение задания №6

```
(defun swap_to_right (lst)
(setf nl (without_last lst))
(cons l nl))

(defun without_last (lst)
(if (cdr lst)
(cons (car lst) (without_last (cdr lst)))
((lambda () (setf l (car lst)) nil))
))
-----------------------------------------
(defun swap_to_left (lst)
(append_into_end (cdr lst) (car lst)))

(defun append_into_end (lst val)
(if (null lst)
(list val)
(cons (car lst) (append_into_end (cdr lst) val))
))
```

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

Листинг 7: Решение задания №7

```
(defun check_existance (el lst)
(cond
((null lst) NIL)
((and (eql (car el) (caar lst)) (eql (cadr el) (cadar lst)) T))
(T (check_existance el (cdr lst)))))

(defun add_element (el lst)
(cond
((check_existance el lst) lst)
(T (cons el lst))))
```

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,
6) элементы списка – любые объекты

а) все элементы списка – числа

Листинг 8: Решение задания №8а

```
(defun mult_only_numbers (num lst)
(cons (* num (car lst)) (cdr lst)))
```

б) элементы списка – любые объекты

Листинг 9: Решение задания №8б

```
(defun mul_first_num (num lst)
(cond
((numberp (car lst)) (cons (* num (car lst)) (cdr lst) ))
((numberp (cadr lst)) (list (car lst) (* num (cadr lst)) (caddr lst) ))
((numberp (caddr lst)) (list (car lst) (cadr lst) (* num (caddr lst))))
(T lst)))
```

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)).

Листинг 10: Решение задания №9

```
(defun check_borders (x1 x2 el) (and (> el x1) (< el x2)))

(defun select_between (lst x1 x2)
(cond
((null lst) NIL)
( (check_borders x1 x2 (car lst)) (cons (car lst) (select_between (cdr lst)
   x1 x2)) )
(T (select_between (cdr lst) x1 x2))))
```

Контрольные вопросы

1. Структурораз рушающие и не разрушающие структуру списка функции.

Ответ. Функции, реализующие опреации со списками, делятся на две группы:

1. не разрушающие структуру функции; данные функции не меняют переданный им объект-аргумент, а создают копию, с которой в дальнейшем производят необходимые преобразования; к таким функциям относятся: append, reverse, last, nth, nthcdr, length, remove, subst и др.

2. структурораз рушающие функции; данные функции меняют сам объект-аргумент, из-за чего теряется возможность работать с исходным списком; чаще всего имя структурораз рушающих функций начинается с префикса -n: nreverse, nconc, nsubst и др.

Обычно в Lisp существуют функции-дубли, которые реализуют одно и то же преобразование, но по разному (с сохранением структуры и без): apppend/nconc, reverse/nreverse и т.д.

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

Ответ. Функция **cons** - чисто математическая, она принимает ровно 2 аргумента, создает бианрный узел и расставляет указатели (car - на первый аргумент, cdr - на второй). В результате работы функции может получится как точечная пара, так и список (зависит от второго аргумента).

Функция **list** - это форма, она принимает произвольное количество аргументов и создает из них список. В отличии от функции cons, list создает столько бинарных узлов, сколько передано ей аргументов, и связывает их вместе. Результатом работы данной функции всегда будет список.

Функция **append** также является формой. Она принимает на вход произвольное число аргументов. Для всех аргументов, кроме последнего, эта функция создает копию, ссылая при этом последний элемент каждого списка аргумента на первый элемент следующего по порядку списка аргумента. В результате работы функции append может получится как список, так и точечная пара (зависит от последнего аргумента).

Итого: **cons** создает однин бинарный узел, **list** создает столько бинарных узлов, сколько передано аргументов, **append** создает копии всех бинарных узлов для каждого из аргументов, исключая последний аргумент.