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 Цветков И.А.

Группа ИУ7-63Б

Оценка (баллы)

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

Москва — 2022 г.

1 Практические задания

1.1 Задание 1

Условие: написать функцию, которая по своему списку-аргументу `lst` определяет является ли он палиндромом (то есть равны ли `lst` и `(reverse lst)`)

```
(defun remove_last (lst)
    (reverse (cdr (reverse lst)))
)


(defun is_equal_first(lst1 lst2)
    (if (eq (first lst1) (first lst2)) T Nil)
)


(defun palindrom (lst)
    (cond ((eq (cdr lst) Nil) T)
        ((is_equal_first lst (reverse lst)) (palindrom (remove_last
           (cdr lst))))
        (T Nil)
    )
)
```

1.2 Задание 2

Условие: написать 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 (lst1 lst2)
    (cond ((and (null lst1) (null lst2)) T)
        ((or (null lst1) (null lst2)) Nil)
        (T (set_equal (remove (car lst1) lst1) (remove (car lst1)
           lst2)))
```

```
    )
)
```

1.3 Задание 3

Условие: напишите свои необходимые функции, которые обрабатывают таблицу из 4-х точечных пар: (страна . столица), и возвращают по стране - столицу, а по столице — страну.

```
(defvar table
    (list (cons "Country1" "Capital1")
        (cons "Country2" "Capital2")
        (cons "Country3" "Capital3")
        (cons "Country4" "Capital4")
    )
)


(defun get_capital (table country)
    (cond ((null table) Nil)
        ((equal (caar table) country) (cdar table))
        (T (get_capital (cdr table) country))
    )
)


(defun get_country (table capital)
    (cond ((null table) Nil)
        ((equal (cdar table) capital) (caar table))
        (T (get_country (cdr table) capital))
    )
)
```

1.4 Задание 4

Условие: напишите функцию `swap-first-last`, которая переставляет в списке-аргументе первый и последний элементы.

```
1
2 (defun remove_last (lst)
3     (reverse (cdr (reverse lst)))
4 )
5
6 (defun swap_first_last (lst)
7     (cons (first (reverse lst)) (append (cdr (remove_last lst))
        (list (first lst))))
8 )
```

1.5 Задание 5

Условие: напишите функцию `swap-two-ellement`, которая переставляет в списке-аргументе два указанных своими порядковыми номерами элемента в этом списке.

```
1 (defun my_replace (lst pos elem)
2     (setf (nth pos lst) elem)
3     lst
4 )
5
6
7 (defun swap_two_element (lst pos1 pos2)
8     (let ((tmp (nth pos1 lst)))
9          (my_replace lst pos1 (nth pos2 lst))
10         (my_replace lst pos2 tmp))
11 )
```

1.6 Задание 6

Условие: напишите две функции, `swap-to-left` и `swap-to-right`, которые производят одну круговую перестановку в списке-аргументе влево и вправо, соответственно.

```
1 (defun remove_last (lst)
2     (reverse (cdr (reverse lst)))
3 )
```

```
4
5  (defun swap_to_left (lst)
6      (append (cdr lst) (list (first lst)))
7  )
8
9
10 (defun swap_to_right (lst)
11     (append (list (first (reverse lst))) (remove_last lst))
12 )
```

1.7 Задание 7

Условие: напишите функцию, которая добавляет к множеству двухэлементных списков новый двухэлементный список, если его там нет.

```
1  (defun is_exist (lst elem)
2      (cond ((null lst) Nil)
3            ((and (= (caar lst) (car elem)) (= (cadar lst) (cadr
                 elem))) T)
4            (T (is_exist (cdr lst) elem))
5      )
6  )
7
8  (defun add_elem (lst elem)
9      (if (is_exist lst elem) lst (append lst (list elem)))
10 )
```

1.8 Задание 8

Условие: напишите функцию, которая умножает на заданное число-аргумент первый числовой элемент списка из заданного 3-х элементного списка-аргумента, когда все элементы списка — числа; элементы списка – любые объекты.

```
1  ; a)
2  (defun mul_first (num lst)
3      (cond ((null lst) nil)
4            ((numberp (car lst)) (setf (car lst) (* (car lst) num)))
5            (T (mul_first num (cdr lst)))
```

```
    )
    lst
)

; 6)
(defun mul_first (num lst)
    (cond ((null lst) Nil)
          ((symbolp (car lst)) (cons (car lst) (f (cdr lst) num)))
          ((listp (car lst)) (cons (f (car lst) num) (f (cdr lst)
             num)))
          (T (cons (* num (car lst)) (f (cdr lst) num)))
    )
)
```

1.9 Задание 9

Условие: 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)).

```

(defun check_border (num a b)
    (and (> num a) (< num b))
)


(defun select_between_rec (lst border_1 border_2 res_lst)
    (let ((elem (car lst)))
         (cond ((null lst) res_lst)
               ((check_border elem border_1 border_2)
                (select_between_rec (cdr lst) border_1 border_2
                   (append res_lst (list elem))))
               (T (select_between_rec (cdr lst) border_1 border_2
                   res_lst))
         )
    )
)

```

```
18
19 (defun select_between (lst border_1 border_2)
20     (let ((res_lst ()))
21         (select_between_rec lst border_1 border_2 res_lst)
22     )
23 )
```

2 Ответы на вопросы к лабораторной работе

2.1 Структуроразрушающие и не разрушающие структуру списка функции

Не разрушающие структуру списка функции – функции, которые не меняют объект-аргумент, а создают копию.

- `length` – возврат длины списка по верхнему уровню;
- `remove` – удаляет элемент по значению (использует `eql`, но можно передать другую функцию через специальный параметр: `:test 'equal`);
- `append` – объединяет списки. Это форма, можно передать больше двух аргументов. Создает копию для всех аргументов, кроме последнего;
- `reverse` – возвращает копию исходного списка, элементы которого переставленыв обратном порядке (только верхний уровень);
- `last` – проход по верхнему уровню и возврат последней списковой ячейки;
- `nth` – возврат указателя от n-ой списковой ячейки, нумерация с нуля.;
- `nthcdr` – возврат n-го хвоста;
- `member` – присутсвует ли эелмент в списке. Возвращает остаток списка, начиная с `car` указателя списковой ячейки;
- `subst` – заменяет все элементы списка, которые равны второму переданному элементу-аргументу на другой первый элемент-аргумент (по умолчанию для сравнения используется функция `eql`).

Структуроразрушающие функции – функции изменяют сам объект, при этом возврат к исходному невозможен.

- `nconc` – аналогично `append`, но не делает коипю, а разрушает структуру;
- `nsubst` – аналогично `subst`, но не создает копию;
- `nreverse` – аналогично `reverse`, но не создает копию.

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

`cons` – имеет фиксированное количество аргументов (два). В случае, когда аргументами являются атомы создает точечную пару. В случает, когда первый аргумент атом а второй список, атом становится головой, а второй аргумент (список) становится хвостом.

`list` – не имеет фиксированное количество аргументов. Создает список, у которого голова - это первый аргумент, хвост - все остальные аргументы.

`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.

`nconc` – аналогично `append`, но не создает копию (разрушает структуру списка).