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 Козлова И.В.

Группа ИУ7-62Б

Оценка (баллы)

Преподаватель Толпинская Н.Б.

Преподаватель Строганов Ю.В.

Москва — 2022 г.

Практические вопросы

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

```
(defun palindrome-p (list)
    (let* ((length (length list))
        (half-length (ceiling length 2))
        (tail (nthcdr half-length list))
        (reversed-head (nreverse (butlast list half-length))))
        (equal tail reversed-head)))


(defun is-palindrome-2 (lst)
    (equalp lst (reverse lst)))
```

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

```
(defun my-subsetp (set1 set2)
    (reduce
        #'(lambda (acc1 set1-el)
            (and acc1 (reduce
                #'(lambda (acc2 set2-el)
                    (or acc2 (= set2-el set1-el)))
                set2 :initial-value Nil)))
    set1 :initial-value T))

(defun set-equal (set1 set2)
    (if (= (length set1) (length set2))
        (and (my-subsetp set1 set2) (my-subsetp set2 set1))
        Nil))
```

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

```
(defun my-assoc (key table)
    (cond ((null table) nil)
        ((equal key (caar table)) (cdar table))
        (T (my-assoc key (cdr table))))))

                    (or acc2 (= set2-el set1-el))) set2 :initial-value Nil)))
    set1 :initial-value T))

(defun set-equal (set1 set2)
    (cond ((= (length set1) (length set2))
            (and (my-subsetp set1 set2) (my-subsetp set2 set1)))
        (T Nil)))
```

```
; столицу
(defun my-assoc (key table)
    (cond ((null table) nil)
        ((equal val (caar table)) (cdar table))
        (T (my-rassoc val (cdr table)))))


; страну
(defun my-rassoc (val table)
    (cond ((null table) nil)
        ((equal val (cdar table)) (caar table))
        (T (my-rassoc val (cdr table)))))

; с помощью стандартных функций
(defun find-capital (key table)
    (cdr (assoc key table)))

(defun find-country (key table)
    (car (rassoc key table)))
```

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

```
(defun f1 (lst)
    (reverse (cdr (reverse lst))) )

(defun swap-first-last (lst)

    (append (append (last lst) (cdr (f1 lst)))
            (cons (car lst) Nil)) )

(defun swap-first-last2 (lst)
    (append (append (last lst) (butlast (cdr lst) 1))
            (cons (car lst) Nil))) )
```

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

```
(defun cut-list (lst n l)
    (cond ((zerop l) nil)
        ((and (= n 1) (> l 0)) (cons (car lst)
                (cut-list (cdr lst) 1 (- l 1))))
        (t (cut-list (cdr lst) (- n 1) l))))

(defun task (lst p q)
    (cond ((= p q) lst)
        ((> p q) (task lst q p))
        (t  (let* ((ls (length lst))
                   (l (cut-list lst 1 (- p 1)))
                   (m (cut-list lst p (- q p)))
                   (r (cut-list lst q (- ls q -1))))
            (append l (list (car r)) (cdr m) (list (car m)) (cdr r))))))
```

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

```
(defun rot-left(n l)
    (append (nthcdr n l) (butlast l (- (length l) n))))

(defun rot-right(n l)
    (rot-left (- (length l) n) l))
```

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)).

```
(defun find-elements (lst left right)
    (remove-if #'(lambda (x) (null x))
        (mapcar #'(lambda (x) (cond (< left x right) x)) lst)))

(defun find-min (lst)
    (setf temp (car lst))
    (mapcar #'(lambda (x)
        (cond (> temp x) (setf temp x))) lst) temp)

(defun set-element (new old lst)
    (cond ((= (car lst) old) (rplaca lst new))
        (t (set-element new old (cdr lst)))) lst)

(defun my-sort (lst)
    (maplist #'(lambda (x)
        (and (setf temp (find-min x))
            (set-element (car x) temp x)
            (rplaca x temp))) lst) lst)

(defun select-between (lst b1 b2)
    (cond ((null lst) nil)
        ((not (and (numberp b1) (numberp b2))) nil)
        ((= b1 b2) nil)
        ((> b1 b2)(my-sort (find-elements lst b2 b1)))
        ((> b2 b1)(my-sort (find-elements lst b1 b2)))))
```