



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 Толплинская Н.Б.

Преподаватель Строганов Ю.В.

Практические вопросы

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

```
1 (defun palindrome-p (list)
2   (let* ((length (length list))
3         (half-length (ceiling length 2))
4         (tail (nthcdr half-length list))
5         (reversed-head (nreverse (butlast list half-length))))
6     (equal tail reversed-head)))
7
8 (defun is-palindrome-2 (lst)
9   (equalp lst (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

```
1 (defun my-subsetp (set1 set2)
2   (reduce
3     #'(lambda (acc1 set1-el)
4         (and acc1 (reduce
5               #'(lambda (acc2 set2-el)
6                   (or acc2 (= set2-el set1-el)))
7               set2 :initial-value Nil)))
8     set1 :initial-value T))
9
10 (defun set-equal (set1 set2)
11   (if (= (length set1) (length set2))
12       (and (my-subsetp set1 set2) (my-subsetp set2 set1))
13       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 .

```
1 (defun my-assoc (key table)
2   (cond ((null table) nil)
3         ((equal key (caar table)) (cdar table))
4         (T (my-assoc key (cdr table)))))
5
6 ; страну
7 (defun my-rassoc (val table)
8   (cond ((null table) nil)
9         ((equal val (cdar table)) (caar table))
10        (T (my-rassoc val (cdr table)))))
11
12 ; с помощью стандартных функций
13 (defun find-capital (key table)
14   (cdr (assoc key table)))
15
16 (defun find-country (key table)
17   (car (rassoc key table)))
```

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

```
1 (defun fl (lst)
2   (reverse (cdr (reverse lst))))
3
4 (defun swap-first-last (lst)
5   (append (append (last lst) (cdr (fl lst)))
6           (cons (car lst) Nil)))
7
8 (defun swap-first-last2 (lst)
9   (append (append (last lst) (butlast (cdr lst) 1))
10          (cons (car lst) Nil)))
```

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

```
1 (defun cut-list (lst n l)
2   (cond ((zerop l) nil)
3         ((and (= n 1) (> l 0)) (cons (car lst)
4                                       (cut-list (cdr lst) 1 (- l 1))))
5         (t (cut-list (cdr lst) (- n 1) l))))
6
7 (defun task (lst p q)
8   (cond ((= p q) lst)
9         ((> p q) (task lst q p))
10        (t (let* ((ls (length lst))
11                  (l (cut-list lst 1 (- p 1)))
12                  (m (cut-list lst p (- q p)))
13                  (r (cut-list lst q (- ls q -1))))
14            (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.

```
1 (defun rot-left (n l)
2   (append (nthcdr n l) (butlast l (- (length l) n))))
3
4 (defun rot-right (n l)
5   (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)).

```
1 (defun find-elements (lst left right)
2   (remove-if #'(lambda (x) (null x))
3     (mapcar #'(lambda (x) (cond (< left x right) x)) lst)))
4
5 (defun find-min (lst)
6   (setf temp (car lst))
7   (mapcar #'(lambda (x)
8     (cond (> temp x) (setf temp x))) lst) temp)
9
10 (defun set-element (new old lst)
11   (cond ((= (car lst) old) (rplaca lst new))
12     (t (set-element new old (cdr lst)))) lst)
13
14 (defun my-sort (lst)
15   (maplist #'(lambda (x)
16     (and (setf temp (find-min x))
17       (set-element (car x) temp x)
18       (rplaca x temp))) lst) lst)
19
20 (defun select-between (lst b1 b2)
21   (cond ((null lst) nil)
22     ((not (and (numberp b1) (numberp b2))) nil)
23     ((= b1 b2) nil)
24     ((> b1 b2) (my-sort (find-elements lst b2 b1)))
25     ((> b2 b1) (my-sort (find-elements lst b1 b2)))))
```