



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 Криков А.В.

Группа ИУ7-63Б

Оценка (баллы) _____

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

Практические задания

Задание 1

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

```
1 (defun pol(lst)
2   (equal lst (reverse lst)))
```

Задание 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 set-equal (set1 set2)
2   (and (= (length set1) (length set2))
3         (every #'(lambda (elem) (member elem set2 :test #'equal))
4                 set1)
5         (every #'(lambda (elem) (member elem set1 :test #'equal))
6                 set2)))
```

Задание 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 search-country (captl table)
2   (cond ((null table) nil)
3         ((eq captl (cdar table)) (caar table))
4         (t (search-country captl (cdr table)))))
5
6 (defun search-capital (cntry table)
7   (cond ((null table) nil)
8         ((eq cntry (caar table)) (cdar table))
9         (t (search-capital cntry (cdr table)))))
10
11 (defun search-country-rassoc (captl table)
12   (car (rassoc captl table)))
```

```

13
14 (defun search-capital-assoc (cntry table)
15   (cdr (assoc cntry table)))

```

Задание 4

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

```

1 (defun swap-first-last (lst)
2   (let* ((tmp (reverse (cdr lst)))
3          (mid (reverse (cdr tmp))))
4     (cons (car tmp) (append mid (list (car lst))))))

```

Задание 5

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

```

1 (defun swap-two-elements (lst i j)
2   (if (or (> i (length lst)) (> j (length lst)))
3       lst
4       (let ((fst (nth i lst))
5             (snd (nth j lst)))
6         (progn (setf (nth i lst) snd) (setf (nth j lst) fst) lst))))

```

Задание 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 swap-to-left (lst)
2   (let* ((tmp (cdr lst))
3          (fml (car lst)))
4     (append tmp (list fml))))
5
6 (defun swap-to-right (lst)
7   (let* ((tmp (reverse (cdr (reverse lst))))
8          (fml (car (reverse lst))))
9     (cons fml tmp)))

```

Задание 7

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

```

1 (defun add-list(src dest)
2   (if (some #'(lambda (pair) (equal dest pair)) src)
3       src
4       (append src (list dest))))

```

Задание 8

Напишите функцию, которая умножает на заданное число-аргумент первый числовой элемент списка из заданного 3-х элементного списка-аргумента, когда:

1. все элементы списка — числа,
2. элементы списка — любые объекты.

```

1 ;; (a)
2 (defun multiply-lst-num (lst mul)
3   (mapcar #'(lambda (elem) (* elem mul)) lst))
4
5 ;; (b)
6 (defun multiply-lst (lst mul)
7   (mapcar #'(lambda (elem) (cond ((listp elem) (multiply-lst elem mul))
8                                ((numberp elem) (* elem mul))
9                                (t elem)))
10  lst))

```

Задание 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 select-between (lst left right)
2   (sort (reduce #'(lambda (res el)
3     (if (and (> el left) (< el right))
4         (cons el res)
5         res)) lst :initial-value ()) #'<))

```

Контрольные вопросы

1. Структуроразрушающие и не разрушающие структуру списка функции.

Не разрушающие структуру функции не меняют сам объект-аргумент, а создают его копию. Например, reverse, append. Разрушающие структуру функции меняют объект-аргумент, и получить исходный уже невозможно. Такие функции начинаются с префикса n: ncons, nreverse.

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

1. `cons` конструирует точечную пару или список, в зависимости от второго аргумента. Является чистой функцией, принимает два аргумента;
2. `list` является формой, принимает произвольное количество аргументов и составляет из них список. Результатом работы всегда является список.
3. `append` является формой, принимает произвольное количество аргументов, создает копию для всех, кроме последнего, при этом последний элемент каждого списка-аргумента ссылается на первый элемент следующего по порядку списка-аргумента (или его копию, в зависимости от расположения).
4. `nconc` возвращает список с элементами из всех списков-аргументов (по порядку). Принцип работы: устанавливает `cdr` последней ячейки каждого списка в начало следующего списка. Последний аргумент может быть объектом любого типа. Вызванная без аргументов, возвращает `nil`.