



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»

Студент _____ Маслова Марина Дмитриевна

Группа _____ ИУ7-63Б

Оценка (баллы) _____

Преподаватель _____ Толпинская Наталья Борисовна

Преподаватель _____ Строганов Юрий Владимирович

2022 г.

1 Практические задания

1.1 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 is-palindrome (lst)
2   (equal lst (reverse lst)))
```

1.2 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-equal1 (set1 set2)
2   (and (subsetp set1 set2) (subsetp set2 set1)))
```

1.3 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 get-capital (table country)
2   (cond ((null table) Nil)
3         ((equal (caar table) country) (cadr table))
4         (T (get-capital (cdr table) country))))
```

```
1 (defun get-country (table capital)
2   (cond ((null table) Nil)
3         ((equal (cadr table) capital) (caar table))
4         (T (get-country (cdr table) capital))))
```

1.4 Задание №4

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

```

1 (defun swap-first-last (lst)
2   (cond ((null (cdr lst)) lst)
3         (T (cons (car (last lst))
4                   (reverse (cons (car lst) (cdr (reverse (cdr lst))))))))))

```

1.5 Задание №5

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

```

1 (defun swap-two-element (lst num1 num2)
2   (let ((len (length lst)))
3     (and (< num1 len) (< num2 len)
4          (let ((fst-el (nth num1 lst))
5                (snd-el (nth num2 lst)))
6            (setf (nth num1 lst) snd-el)
7            (setf (nth num2 lst) fst-el))
8            lst)))

```

1.6 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   (append (cdr lst) (cons (car lst) Nil)))

```

```

1 (defun swap-to-right (lst)
2   (cons (car (last lst)) (reverse (cdr (reverse lst)))))

```

1.7 Задание №7

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

```

1 (defun add-two-el-list (list-set lst)
2   (cond ((null list-set) (list lst))
3         ((equal (car list-set) lst) list-set)
4         (T (cons (car list-set) (add-two-el-list (cdr list-set) lst)))))

```

1.8 Задание №8

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

1. все элементы списка — числа,

```
1 (defun mul-first-nums (lst num)
2   (and lst (setf (car lst) (* (car lst) num)) lst))
```

2. элементы списка — любые объекты.

```
1 (defun mul-first-num (lst num)
2   (cond ((null lst) lst)
3         ((numberp (car lst)) (cons (* (car lst) num) (cdr lst)))
4         (t (cons (car lst) (mul-first-num (cdr lst) num)))))
```

1.9 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).

```
1 (defun select-between (lst begin end)
2   (cond ((null lst) Nil)
3         ((not (< begin (car lst) end))
4          (select-between (cdr lst) begin end))
5         (t (cons (car lst) (select-between (cdr lst) begin end)))))
6
7 (defun sort-select-between (lst begin end)
8   (sort (select-between lst begin end) #'<))
```

2 Теоретические вопросы

2.1 Структуроразрушающие и не разрушающие структуру списка функции

Структуроразрушающие функции — функции, после использования которых теряется возможность работы изначальными списками. Эти функции изменяют сам объект. Чаще всего такие функции начинаются с префикса `n` (например, `ncons`, `nreverse` и т. п.).

Функции, не разрушающие структуру списка — функции, после использования которых сохраняется возможность работы с изначальными списками. Эти функции не изменяют сам объект, а создают копии (например, `append`, `reverse`, `length` и т. п.).

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

`cons` создает списковую ячейку и расставляет указатели (`car`-указатель — на первый аргумент, `cdr`-указатель — на второй аргумент), если второй аргумент — список, то возвращает список, если нет — точечную пару.

`list` создает столько списковых ячеек, сколько аргументов, возвращает список.

`append` создает копии всех элементов кроме последнего и расставляет `cdr`-указатели последних элементов копий на следующие копии/аргумент.

`ncons` аналогична `append` только работает не с копиями, а с исходными списками.