

## 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 №10 по курсу "Функциональное и логическое программирование"

Тема _	Вложенные рекурсия и функционалы
Студе	нт Пересторонин П.Г.
Групп	а ИУ7-63Б
Оценк	a
Препо	даватель Толпинская Н. Б.

### Оглавление

1	Зад	дания	2
	1.1	Написать функцию, которая по своему аргументу-списку lst	
		определяет, является ли он полиндромом (то есть равны ли	
		lst и (reverse lst))	2
	1.2	Написать предикат set-equal, который возвращает t, если	
		два его множества-аргумента содержат одни и те же элемен-	
		ты, порядок которых не имеет значения	2
	1.3	Напишите необходимые функции, которые обрабатывают таб-	
		лицу из точечных пар: (страна . столица), и возвращают	
		по стране столицу, а по столице — страну	2
	1.4	Напишите функцию swap-first-last, которая переставляет	
		в списке аргументе первый и последний элементы	3
		1.4.1 Разрушающая структуру	3
		1.4.2 Не разрушающая структуру	3
	1.5	Напишите функцию swap-two-ellement, которая перестав-	
		ляет в списке-аргументе два указанных своими порядковыми	
		номерами элемента в этом списке	4
	1.6	Разрушающая структуру	4
	1.7	Не разрушающая структуру	4
	1.8	Напишите две функции, swap-to-left и swap-to-right, ко-	
		торые производят круговую перестановку в списке-аргументе	
		влево и вправо, соответственно	5
2	Отн	веты на вопросы к лабораторной работе	6
	2.1	Способы определения функций	6
	2.2	Варианты и метолы молификации списков	6

#### 1 Задания

1.1 Написать рекурсивную версию (с именем rec-add) вычисления суммы чисел заданного списка

```
(defun rec-add-internal (lst acc)
(cond ((null lst) acc)
(T (rec-add-internal (cdr lst) (+ acc (car lst)))))
(defun rec-add (lst)
(rec-add-internal lst 0))
```

1.2 Написать рекурсивную функцию с именем rec-nth функции nth

```
(defun rec-nth (n lst)
(and lst (cond ((zerop n) (car lst))
(T (rec-nth (- n 1) (cdr lst))))))
```

1.3 Написать рекурсивную функцию alloddr, которая возвращает t, когда все элементы списка нечётные

```
(defun alloddr (lst)
(or (null lst)
(and (oddp (car lst))
(alloddr (cdr lst)))))
```

1.4 Написать рекурсивную функцию, относящуюся к хвостовой рекурсии с одним тестом завершения, которая возвращает последний элемент списка-аргумента

```
(defun last-rec (lst)
(cond ((null (cdr lst)) (car lst))
(T (last-rec (cdr lst))))
```

- 1.5 Написать рекурсивную функцию, относящуюся к дополняемой рекурсии с одним тестом завершения, которая вычисляет сумму всех чисел от 0 до *n*-ого аргумента функции
- 1.5.1 От n-аргумента функции, до последнего >=0
- 1.5.2 От n-аргумента функции до m-аргумента с шагом d
- 1.6 Написать рекурсию, которая возвращает последнее нечетное число из числового списка, возможно создавая некоторые вспомогательные функции

<sup>(</sup>defun last-odd-internal (lst cur-odd)
(cond ((null lst) cur-odd)

```
(T (cond ((oddp (car lst)) (last-odd-internal (cdr lst) (car lst)))

(T (last-odd-internal (cdr lst) cur-odd)))))

(defun last-odd (lst)

(last-odd-internal lst nil))
```

1.7 Используя cons-дополняемую рекурсию с одним тестом завершения, написать функцию, которая получает как аргумент список чисел, а возвращает список квадратов этих чисел в том же порядке

```
(defun cons-square (lst)
(and lst (cons ((lambda (x) (* x x)) (car lst))
(con-square (cdr lst)))))
```

1.8 Написать функцию с именем select-odd, которая из заданного списка выбирает все нечетные числа

#### Варианты:

- 1. select-even;
- 2. вычисляет сумму всех нечетных чисел (sum-all-odd) или сумму всех четных чисел из заданного списка.

```
; reverses the list
(defun my-filter-reducer (filter reducer lst init-el)

(reduce #'(lambda (acc x)

(cond ((funcall filter x) (funcall reducer acc x))

(T acc)))

(cons init-el lst)))

(defun my-filter (func lst)
```

```
(reverse (my-filter-reducer func #'(lambda (acc x) (cons x acc)) lst nil)))

(defun select-odd (lst)
   (my-filter #'oddp lst))

(defun select-even (lst)
   (my-filter #'evenp lst))

(defun sum-all-odd (lst)
   (my-filter-reducer #'oddp #'+ lst 0))

(defun sum-all-even (lst)
   (my-filter-reducer #'evenp #'+ lst 0))
```

# 1.9 Создать и обработать смешанный структурированный список с информацией:

- ФИО;
- зарплата;
- возраст;
- категория (квалификация).

Изменить зарплату в зависимости от заданного условия, и подсчитать суммарную зарплату. Использовать композиции функций.