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

Студент Кононенко С.С.

Группа ИУ7-63Б

Оценка (баллы) _____

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

Москва — 2021 г.

Задание 1

Постановка задачи. Написать рекурсивную версию вычисления суммы чисел заданного списка.

Решение.

Листинг 1 – Решение задания 1

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

Задание 2

Постановка задачи. Написать рекурсивную версию функции `nth`.

Решение.

Листинг 2 – Решение задания 2

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

Задание 3

Постановка задачи. Написать рекурсивную функцию `alloddp`, которая возвращает `T`, когда все элементы спсика нечетные.

Решение.

Листинг 3 – Решение задания 3

```
(defun all-odd-p (lst)
    (cond ((null lst) T)
    ((evenp (car lst)) Nil)
    (T (all-odd-p (cdr lst)))))
```

Задание 4

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

Решение.

Листинг 4 – Решение задания 4

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

Задание 5

Постановка задачи. Написать рекурсивную функцию, относящуюся к дополняемой рекурсии с одним тестом завершения, которая вычисляет сумму всех чисел от 0 до n-ого аргумента функции.

Решение.

Листинг 5 – Решение задания 5

```
(defun make-sum-zero-n (lst n)
    (cond ((or (null lst) (zerop n)) 0)
    (T (+ (car lst) (make-sum-zero-n (cdr lst) (- n 1))))))
```

Задание 6

Постановка задачи. Написать рекурсивную функцию, которая возвращает последнее нечетное число из числового списка.

Решение.

Листинг 6 – Решение задания 6

```
(defun internal (lst el)
    (cond ((null lst) el)
    (T (cond ((oddp (car lst)) (internal (cdr lst) (car lst)))
        (T (internal (cdr lst) el))))))
(defun make-last-odd (lst)
```

```
6     (internal lst Nil))
```

Задание 7

Постановка задачи. 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.

Решение.

Листинг 7 – Решение задания 7

```
1 (defun make-square (lst)
2     (cond ((null lst) Nil)
3     (T (cons ((lambda (el) (* el el)) (car lst)) (make-square (cdr lst))))))
```

Задание 8

Постановка задачи. Написать функцию, которая из заданного списка выбирает все нечетные числа.

Решение.

Листинг 8 – Решение задания 8

```
1 (defun make-odd (lst)
2     (mapcan #'(lambda (el) (if (oddp el) (list el))) lst))
```

Задание 9

Постановка задачи. Создать и обработать смешанный структурированный список, хранящий информацию о работниках (ФИО, зарплата, возраст, квалификация). Изменить зарплату в зависимости от заданного условия и подсчитать суммарную зарплату.

Решение.

Листинг 9 – Решение задания 9

```
1 (setf people (list
```

```
    (list
        (cons 'Initials "Romanov Alexey")
        (cons 'Salary 35000)
        (cons 'Age 20)
        (cons 'Place "Tarantool Engineering"))
    (list
        (cons 'Initials "Pavel Perestoronin")
        (cons 'Salary 100500)
        (cons 'Age 20)
        (cons 'Place "Qoolo"))
    (list
        (cons 'Initials "Mikhail Nitenko")
        (cons 'Salary 500000)
        (cons 'Age 21)
        (cons 'Place "Huawei"))
    (list
        (cons 'Initials "Dmitry Yakuba")
        (cons 'Salary 1)
        (cons 'Age 20)
        (cons 'Place "DeadBrains"))
    )
)
(defun make-salaries-sum (lst)
    (reduce #'(lambda (acc x) (+ acc (cdr (assoc 'Salary x)))) lst :initial-value 0))
(defun internal (func el)
    (setf (cdr (assoc 'Salary el)) (funcall func (cdr (assoc 'Salary el)))))
(defun change-salary (lst cp sf)
    (mapcar #'(lambda (el) (if (funcall cp el) (internal sf el) (cdr (assoc 'Salary
        el)))) lst))
```