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 дисциплине «Функциональное и логическое программирование»

Тема Вложенная рекурсия и функционалы

Студент Романов А.В.

Группа ИУ7-63Б

Оценка (баллы) _____

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

Москва — 2021 г.

Задание №1

Постановка задачи

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

Решение

```
(defun rec-add (lst)
  (defun wrapper (lst acc)
    (if (null lst) acc
      (wrapper (cdr lst) (+ acc (car lst)))))
  (wrapper lst 0))
```

Задание №2

Постановка задачи

Написать рекурсивную функцию с именем `rec-nth` функции `nth`.

Решение

```
(defun rec-nth (n lst)
  (if (zerop n)
    (car lst)
    (rec-nth (- n 1) (cdr lst))))
```

Задание №3

Постановка задачи

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

Решение

```
(defun alloddr (lst)
  (if (null lst) T
    (and (oddp (car lst)) (alloddr (cdr lst)))))
```

Задание №4

Постановка задачи

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

Решение

```
(defun rec-last (lst)
  (if (null (cdr lst))
    (car lst)
    (rec-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

```
(defun rec-sum-n (n lst)
  (if (or (zerop n) (null lst)) 0
    (+ (car lst) (sum-n (- n 1) (cdr lst)))))
```

Задание №6

Постановка задачи

Написать рекурсию, которая возвращает последнее нечетное число из числового списка, возможно создавая некоторые вспомогательные функции.

Решение

```
(defun rec-last-odd (lst)
  (defun wrapper (lst cur)
    (if (null lst) cur
      (if (oddp (car lst))
        (wrapper (cdr lst) (car lst))
        (wrapper (cdr lst) cur))))
  (wrapper 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

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

Задание №8

Постановка задачи

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

Решение

```
(defun select-odd (lst)
  (reduce #'(lambda (acc x)
    (if (oddp x)
      (append acc (cons x Nil))
      acc))
  lst :initial-value ()))

(defun select-sum-odd (lst)
  (reduce #'(lambda (acc x)
```

```
    (if (oddp x)
      (+ acc x)
      acc))
  lst))
```

Задание №9

Постановка задачи

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.

Решение

```
(setf people (list
  (list
    (cons 'FIO "Sergey Kononenko")
    (cons 'Salary 30000)
    (cons 'Age 21)
    (cons 'Category "Tarantool Presale"))
  (list
    (cons 'FIO "Pavel Perestoronin")
    (cons 'Salary 100500)
    (cons 'Age 20)
    (cons 'Category "Qoolo developer"))
  (list
    (cons 'FIO "Mikhail Nitenko")
    (cons 'Salary 3000)
    (cons 'Age 32)
    (cons 'Category "Superflex C++ developer"))
  (list
    (cons 'FIO "Dmitry Yakuba")
    (cons 'Salary 500)
    (cons 'Age 58)
    (cons 'Category "Kotlin)"))
  )
)
```

```
(defun sum-salaries (lst)
  (reduce #'(lambda (acc x)
    (+ acc (cdr (assoc 'Salary x))))
  lst :initial-value 0))

(defun change-salaries (lst changep salary-func)
  (defun wrapper (salary-func x)
    (setf (cdr (assoc 'Salary x)) (funcall salary-func (cdr (assoc 'salary x
       )))))

  (mapcar #'(lambda (x)
    (if (funcall changep x)
      (wrapper salary-func x)
      (cdr (assoc 'Salary x))))
    lst))

(change-salaries people
        #'(lambda (x) (< (cdr (assoc 'Age x)) 30)) ;; predicate
        #'(lambda (x) (+ x 15))) ;; change func
```