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 (ИУ7)**

ОТЧЕТ

по лабораторной работе № 10

Название: Вложенные рекурсия и функционалы.

Дисциплина: Функциональное и логическое программирование

Студент ИУ7-63Б _____ В.П. Федоров
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Н.Б. Толпинская
(Подпись, дата) (И.О. Фамилия)

Москва, 2021

Задача №8.
Написать рекурсивную версию вычисления сумму чисел заданного списка.

```
(defun list-sum(lst)
        (cond
                ((null (cdr lst)) (car lst))
                (t (+ (car lst) (list-sum(cdr lst))))))



(defun list-sum(lst)
        (cond
                ((null lst)) 0
                (t (+ (car lst) (list-sum (cdr lst))))))
```

Задача №9.
Написать рекурсивную версию функции nth.

```
(defun my-nth(n lst)
        (cond
                ((> n (length lst)) nil)
                ((< n 0) 'error)
                ((= n 0) (car lst))
                (t (my-nth (cdr lst) (- n 1)))))
```

```
* (my-nth '(1 2 3 4 5 6) '3)
4
* (my-nth '(1 2 3 4 5) '10)
NIL
* (my-nth '(1 2 3) '-3)
ERROR
* (my-nth '(1 2 3) '0)
1
```

Задача №10.
Написать рекурсивную функцию allodd, которая возвращает t, когда все элементы списка нечетные.

```
(defun allodd(lst)
        (cond
                ((null (cdr lst)) (if (null (car lst)) Nil T))
                ((oddp (car lst)) (allodd (cdr lst)))
                (t Nil)))
```

```
* (allodd '(1 4 5 5))
NIL

* (allodd '(1 3 5))
T

* (allodd '())
NIL

* (allodd '(2 3 3))
NIL
```

Задача №11.
Написать рекурсивную функцию, относящуюся к хвостовой рекурсии с одним тестом завершения, которая возвращает последний элемент списка-аргумента.

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

```
* (GET-LAST-REC '(1 2 3))
3
* (GET-LAST-REC '(1))
1
* (GET-LAST-REC '())
NIL
```

Задача №12.

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

Вариант 1: от n-аргумента функции до последнего >= 0.

Листинг 12: задача 12

```
(defun get-n-sum(lst n)
        (cond
                ((= n 0) 0)
                ((OR (= n 1) (null (cdr lst))) (car lst))
                (t (+ (car lst) (get-n-sum (cdr lst) (- n 1))))))
```

```
* (GET-N-SUM '(1 2 3) 0)
0
*  (GET-N-SUM '(1 2 3 4 5) 10)
15
*  (GET-N-SUM '(1 2 3 4 5) 3)
6
```

Листинг 13: вариант 1

```
(defun get-n-sum-from(lst n)
        (cond
                ((> n (length lst)) 0)
                ((< n 0) 'error)
                ((> n 0) (get-n-sum-from (cdr lst) (- n 1)))
                ((= n 0) (if (car lst)
                                        (+ (car lst) (get-n-sum-from (cdr lst) n)) 0))))
```

Задача №13.

Написать рекурсивную функцию, которая возвращает последнее нечетное числа из числового списка, возможно создавая некоторые вспомогательные функции.

```
(defun get-last-odd(lst)
        (get-first-odd (reverse lst)))

(defun get-first-odd(lst)
        (cond
```

```
        ((null lst) nil)
        ((oddp (car lst)) (car lst))
        (t (get-first-odd (cdr lst)))))
```

(get-last-odd '(1 2 3 4 5 6))
5

Задача №14.

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

```
(defun get-squares(lst)
    (cond
        ((null lst) nil)
        ((numberp (car lst)) (cons (* (car lst) (car lst)) (get-squares (cdr lst))))
        ((listp (car lst)) (append (get-squares (car lst)) (get-squares (cdr lst))))
        (t (get-squares (cdr lst)))))
```

* (GET-SQUARES '(1 2 3 4))
(1 4 9 16)

Задача №15.

Написать функцию с именем select-odd, которая из заданного списка выбирает все нечетные числа.
Вариант 1: select-even,
Вариант 2: вычисляет сумму всех нечетных чисел или сумму всех четных чисел

Листинг 15: вариант 1

```
(defun select-odd(lst)
    (mapcar #'(lambda (x)
                                        (cond ((oddp x) x))) lst))


(defun select-odd(lst)
    (cond
        ((null lst) nil)
        ((oddp (car lst)) (cons (car lst) (select-odd (cdr lst))))
        (t (select-odd (cdr lst)))))
```

Листинг 16: вариант 2

```
; сумма всех нечетных
(defun select-odd(lst)
	(cond
		((null lst) 0)
		((oddp (car lst)) (+ (car lst) (select-odd (cdr lst))))
		(t (select-odd (cdr lst)))))

; сумма всех четных
(defun select-sum-even(lst)
	(cond
		((null lst) 0)
		((evenp (car lst)) (+ (car lst) (select-sum-even (cdr lst))))
		(t (select-sum-even (cdr lst)))))
```