|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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**

*к лабораторной работе №10*

*По курсу: «Функциональное и логическое программирование»*

Студентка ИУ7-65Б

Оберган Т.М

Преподаватель

Толпинская Н.Б

*Москва, 2020 г.*

**7. Пусть list-of-list список, состоящий из списков. Написать функцию, которая вычисляет сумму длин всех элементов list-of-list, т.е. например для аргумента ((1 2) (3 4)) -> 4.**

; lol – список списков

; sum – сумма длин

(defun list-of-list (lol sum)

(cond

((null lol) sum)

(t (list-of-list (cdr lol) (+ sum (length (car lol)))))))

(defun call\_lol (lol)

(list-of-list lol 0))

(call\_lol '((1 2 3) ((4 4) 5 6))) ; 6

**8. Написать рекурсивную версию (с именем reg-add) вычисления суммы чисел заданного списка.**

**Например: (reg-add (2 4 6)) -> 12**

(defun reg-add (lst sum)

(cond

((null lst) sum)

(t (reg-add (cdr lst) (+ sum (car lst))))))

(defun call-reg (lst)

(reg-add lst 0))

(call-reg '(2 4 6)) ; 12

**9. Написать рекурсивную версию с именем recnth функции nth.**

(defun recnth (n lst)

(cond

((< n 0) nil) ; нужна ли тут проверка? в nth выкидывает ошибку

((= n 0) (car lst))

(t (recnth (- n 1) (cdr lst)))))

(recnth 1 '(0 1 2 3)) ; 1

(recnth 4 '(0 1 2 3)) ; nil

**10. Написать рекурсивную функцию alloddr, которая возвращает t когда все элементы списка нечетные.**

(defun alloddr (lst)

(cond

((null lst) t)

((oddp (car lst)) (alloddr (cdr lst)))

(t nil)))

(alloddr ‘(1 2)) ; nil

(alloddr ‘(1)) ; t

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

(defun my-last (lst)

(cond

((null (cdr lst)) (car lst))

(t (my-last (cdr lst)))))

(my-last ‘()) ; nil

(my-last ‘(1 2 3)) ; 3

**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 п-аргумента функции до последнего >= 0,**

**2) от п-аргумента функции до т-аргумента с шагом d.**

(defun my-sum (start stop step)

(cond

((> stop start) (+ stop (my-sum start (- stop step) step)))

(t 0)))

(my-sum 0 4 2) ; 6

Вариант 1 получается при помощи установки аргумента start в 0, а step в 1.

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

(defun last-odd (lst &optional res)

(cond

((null lst) res)

((oddp (car lst)) (last-odd (cdr lst) (car lst)))

(t (last-odd (cdr lst) res))))

(last-odd ‘(2 4)) ; nil

(last-odd ‘(1 2 3 4)) ; 3

**14. 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 inner-square (lst &optional res)

(cond

((null lst) res)

(t (inner-square (cdr lst) (cons (\* (car lst) (car lst)) res)))))

(defun square-lst (lst)

(reverse (inner-square lst)))

(square-lst '(1 2 3 4)) ; (1 4 9 16)

**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(sum-all-odd) или сумму всех четных чисел (sum-all-even) из заданного списка.**

(defun select-odd (lst &optional res)

(cond

((null lst) res)

((oddp (car lst)) (select-odd (cdr lst) (cons (car lst) res)))

(t (select-odd (cdr lst) res))))

(select-odd '(1 2 3 4)) ; (3 1)

(defun inner-odd-sum (lst sum)

(cond

((null lst) sum)

((oddp (car lst)) (inner-odd-sum (cdr lst) (+ sum (car lst))))

(t (inner-odd-sum (cdr lst) sum))))

(defun sum-all-odd (lst)

(inner-odd-sum lst 0))

(sum-all-odd '(1 2 3 4)) ; 4

(sum-all-odd '(1 2 3 4 4)) ; 4

(sum-all-odd '()) ; 0

(sum-all-odd '(2 4 6)) ; 0

Для реализации функций select-even и sum-all-even нужно заменить oddp на evenp.

**Способы организации повторных вычислений в Lisp:**

Функционалы, рекурсия.

**Что такое рекурсия?**

Рекурсия — это ссылка на определяемый объект во время его определения

**Классификация рекурсивных функций в Lisp**

* простая – один рекурсивный вызов
* первого порядка – рекурсивный вызов встречается несколько раз
* взаимная – используется несколько функций, рекурсивно вызывающих друг друга

**Различные способы организации рекурсивных функций**

* хвостовая – результат формируется на входе в рекурсию
* по нескольким параметрам
* дополняемая рекурсия – используется дополнительная функция вне аргумента вызова
* множественная рекурсия – на одной ветке несколько рекурсивных вызовов

**Способы повышения эффективности реализации рекурсии**.

Рекомендуется формировать результат не на выходе из рекурсии, а на входе в рекурсию, все действия выполняя до ухода на следующий шаг рекурсии