|  |  |
| --- | --- |
|  | **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 №7**

по курсу «Функциональное и логическое программирование»

по теме «Рекурсивные функции»

Студент: Костев Д.И.

Группа: ИУ7-61Б

Преподаватель: Толпинская Н.Б.

*2022 г.*

**Практические задания**

***1.* *Написать хвостовую рекурсивную функцию my-reverse, которая развернет верхний уровень своего списка-аргумента lst.***

Решение:

(defun rec\_reverse (lst res)

(cond

((null lst) res)

(T (rec\_reverse (cdr lst) (cons (car lst) res)))))

(defun my\_reverse (lst) (rec\_reverse lst ()))

Результат работы:

(my\_reverse '(1 2 3 4)) → (4 3 2 1)

***2. Написать функцию, которая возвращает первый элемент списка -аргумента, который сам является непустым списком.***

Решение:

(defun return\_el\_list (lst)

(cond

((null lst))

((and (listp (car lst)) (not (null (car lst)))) (car lst))

(T (return\_el\_list (cdr lst)))))

Результат работы:

(return\_el\_list '(1 () abc (1 2) 3)) → (1 2)

***3. Написать функцию, которая выбирает из заданного списка только те числа, которые больше 1 и меньше 10. (Вариант: между двумя заданными границами)***

Решение:

(defun check\_borders (el) (and (> el 1) (< el 10)))

(defun select\_rec (lst res)

(cond

((null lst) res)

((check\_borders (car lst)) (select\_rec (cdr lst) (cons (car lst) res)))

(T (select\_rec (cdr lst) res))))

(defun select\_between (lst) (select\_rec lst ()))

Результат работы:

(select\_between '(1 2 13 4 15 6)) → (6 4 2)

***4.* *Напишите рекурсивную функцию, которая умножает на заданное число-аргумент все числа из заданного списка-аргумента, когда***

*a) все элементы списка – числа,*

*6) элементы списка – любые объекты*

Решение:

а) (defun mult\_num\_rec (lst num)

(cond

((null lst))

(T (setf (car lst) (\* num (car lst))) (mult\_num\_rec (cdr lst) num))))

(defun mult\_only\_numbers (lst num) (mult\_num\_rec lst num) lst)

Результат работы:

(mult\_only\_numbers '(1 2 3 4) 2) → (2 4 6 8)

б) (defun mult\_var\_rec (lst num)

(cond

((null lst))

((numberp (car lst)) (setf (car lst) (\* num (car lst)))

(mult\_var\_rec (cdr lst) num)) (T (mult\_var\_rec (cdr lst) num))))

(defun mult\_various (lst num) (mult\_var\_rec lst num) lst)

Результат работы:

(mult\_various '(1 abc (2 3) 4 5 d) 4) → (4 ABC (2 3) 16 20 D)

***5. Напишите функцию, select-between, которая из списка-аргумента, содержащего только числа, выбирает только те, которые расположены между двумя указанными границами-аргументами и возвращает их в виде списка (упорядоченного по возрастанию списка чисел (+ 2 балла)).***

Решение:

(defun check\_borders (el a b) (and (> el a) (< el b)))

(defun select\_rec (lst a b res)

(cond

((null lst) res)

((check\_borders (car lst) a b) (select\_rec (cdr lst) a b (cons (car lst) res)))

(T (select\_rec (cdr lst) a b res))))

(defun select\_between (lst a b) (select\_rec lst a b ()))

Результат работы:

(select\_between '(1 2 3 4 5 6) 2 5) → (4 3)

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

*а) одноуровнего смешанного,*

*б) структурированного.*

Решение:

а) (defun add\_1 (lst sum)

(cond

((null lst) sum)

((numberp (car lst)) (add\_1 (cdr lst) (+ sum (car lst))))

(T (add\_1 (cdr lst) sum))))

(defun rec\_add\_1 (lst) (add\_1 lst 0))

Результат работы:

(rec\_add\_1 '(1 2 3 4 5)) → 15

(rec\_add\_1 '(1 a 3 bc 5 cde)) → 9

б) (defun add\_2 (lst sum)

(cond

((null lst) sum)

((numberp (car lst)) (add\_2 (cdr lst) (+ sum (car lst))))

((listp (car lst)) (add\_2 (cdr lst)(+ sum (add\_2 (car lst) 0))))

(T (add\_2 (cdr lst)sum))))

(defun rec\_add\_2 (lst) (add\_2 lst 0))

Результат работы:

(rec\_add\_2 '(1 2 (3 4) (5) () 6 7)) → 28

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

Решение:

(defun get\_el\_by\_pos (n lst ind)

(cond

((null lst))

((= ind n) (car lst))

(T (get\_el\_by\_pos n (cdr lst) (+ ind 1)))))

(defun recnth (n lst) (get\_el\_by\_pos n lst 0))

Результат работы:

(recnth 3 '(1 2 3 4 5 6)) → 4

***8. Написать рекурсивную функцию allodd, которая возвращает t когда все элементы списка нечетные.***

Решение:

(defun all\_odd (lst)

(cond

((null lst) T)

((oddp (car lst)) (all\_odd (cdr lst)))))

Результат работы:

(all\_odd '(1 3 5 7)) → T

(all\_odd '(1 2 3 4 5 6 7)) → NIL

***9.* *Написать рекурсивную функцию, которая возвращает первое нечетное число из списка (структурированного), возможно создавая некоторые вспомогательные функции.***

Решение:

(defun first\_odd (lst)

(cond

((null lst))

((and (numberp (car lst)) (oddp (car lst))) (car lst))

((listp (car lst)) (or (first\_odd (car lst)) (first\_odd (cdr lst))))

(T (first\_odd (cdr lst)))))

Результат работы:

(first\_odd '(2 4 6 7 3 2)) → 7

(first\_odd '(2 4 (2 4) (6 5))) → 5

***10.* *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 square\_rec (lst res)

(cond

((null lst) res)

(T (square\_rec (cdr lst) (append res (cons (\* (car lst) (car lst)) NIL))))))

(defun return\_square (lst) (square\_rec lst ()))

Результат работы:

(return\_square '(1 2 3 4)) → (1 4 9 16)