**Студент: Ятагани Клейди**

**Группа: ИУ7-66и**

**Задание 1**

Дан смешанный, структурированный список (т.е. любой элемент списка может быть: символ, число или список).

Найти сумму четных чисел на всех уровнях заданного списка, лежащих в заданном интервале [a, b].

Добавить найденную сумму к исходному списку, в качестве **К**-ого элемента верхнего уровня или в конец. Если чисел нет, сообщить об этом.

\*Преобразовать исходный список в одноуровневый, сохранив порядок, но удалив числа (для max-х баллов).

!!! Использовать **свои** функции и базисные, основные функции.

!!! Реализовать свои функции **двумя** способами: рекурсивно и с помощью основных функционалов.

!!! Указать назначение своих функций и параметров.

Part 1

Вычисляет сумму четных чисел от a до b

(defun get\_sum\_helper (lst a b sum)

(cond

(

(null lst)

sum

)

(

(listp (car lst))

(get\_sum\_helper (cdr lst) a b (get\_sum\_helper (car lst) a b sum))

)

(

(and

(numberp (car lst))

(evenp (car lst))

(or

(and (>= a (car lst))(<= b (car lst)))

(and (<= a (car lst))(>= b (car lst)))

)

)

(get\_sum\_helper (cdr lst) a b (+ sum (car lst)))

)

(

(get\_sum\_helper (cdr lst) a b sum)

)

)

)

Вспомогательная функция-обертка для вычисления суммы

(defun get\_sum (lst a b)

(get\_sum\_helper lst a b 0))

Проверяет, что в списке есть числа

(defun check\_nums\_helper (lst check)

(cond

(

(or (null lst) check)

check

)

(

(numberp (car lst))

)

(

(listp (car lst))

(check\_nums\_helper (cdr lst) (check\_nums\_helper (car lst) check))

)

(

(check\_nums\_helper (cdr lst) check)

)

)

)

Вспомогательная функция обертка для проверки, что в списке есть числа

(defun check\_nums (lst)

(check\_nums\_helper lst nil))

Функция вставки элемента на k-ую позицию:

(defun insert\_element\_helper (lst k element new\_lst)

(cond

(

(null lst)

new\_lst

)

(

(= k 0)

(insert\_element\_helper (cdr lst) (- k 1) element (cons (car lst) (cons element new\_lst)))

)

(

(insert\_element\_helper (cdr lst) (- k 1) element (cons (car lst) new\_lst))

)

)

)

Вспомогательная функция-обертка для вставки элемента на k-ую позицию:

(defun insert\_element (lst k element)

(reverse (insert\_element\_helper lst k element nil)))

Функция, которая определяет, есть ли в списке числа, если есть, то она вычисляет сумму четных чисел между a и b и вставляет на k-ую позицию списка, иначе выводит сообщение, что чисел нет:

(defun task (lst a b k)

(cond

(

(check\_nums lst)

(insert\_element lst k (get\_sum lst a b))

)

((princ "There are no numbers in the list"))

)

)

Part 2 - Functional

Функция, возвращающая список четных чисел между a и b:

(defun get\_between (lst a b)

(mapcan #'(lambda (x)

(cond

(

(and

(numberp x)

(evenp x)

(or

(and (>= a x)(<= b x))

(and (<= a x)(>= b x))

)

)

(cons x nil)

)

(

(listp x)

(get\_between x a b)

)

)

) lst

)

)

Функция, возвращающая сумму четных чисел между a и b:

(defun get\_sum (lst a b)

(reduce #'+ (get\_between lst a b)))

Функция, возвращающая список из t и nil, соответствующих проверке каждого элемента на число:

(defun is\_nums (lst)

(mapcan #'(lambda (x)

(cond

(

(numberp x)

(cons t nil)

)

(

(listp x)

(is\_nums x)

)

(

(cons nil nil)

)

)

) lst )

)

Функция, проверяющая, есть ли в списке числа:

(defun check\_nums (lst)

(and lst (reduce #'(lambda (a b)

(or a b)

)

(is\_nums lst)

)

)

)

Функция, которая определяет, есть ли в списке числа, если есть, то она вычисляет сумму четных чисел между a и b и вставляет ее в конец списка, иначе выводит сообщение, что чисел нет:

(defun task (lst a b)

(cond

(

(check\_nums lst)

(nconc lst (cons (get\_sum lst a b) nil))

)

(pr

(inc "There are no numbers in the list")

)

) )