*Белоусова Ю.С. ИУ7-61Б*

**РК2 Lisp**

a, b - границы интервала

lst - заданный список

k - номер позиции, куда записать сумму

*Реализация с помощью функционалов.*

Функция, возвращающая список всех элементов, подходящих под условие, сумму которых надо найти.

(defun for\_sum\_nums (a b lst)

(mapcan #'(lambda (el)

(cond ((and (numberp el) (evenp el) (>= el a) (<= el b)) (list el))

((listp el) (for\_sum\_nums a b el)))) lst))

Функция, возвращающая список значений nil и t, в зависимости от того, является ли элемент числом или нет (если чисел нет, то список состоит полностью из nil).

(defun for\_find\_nums (lst)

(mapcan #'(lambda (el)

(cond ((numberp el) (list t))

((listp el) (for\_find\_nums el))

(t (list nil)))) lst))

Основная функция, объединяющая работу всех предыдущих. В ней проверяется, есть ли в исходном списке числа, с помощью reduce и лямбда-функции с предикатом or и полученного списка, состоящего из t и nil, а также суммируются найденные числа из списка с помощью apply. Сумма добавляется в конец списка.

(defun work (a b lst)

(cond ((not (reduce #'(lambda (x y) (or x y)) (for\_find\_nums lst))) (print "no numbers in the list!"))

(t (append lst (list (apply #'+ (for\_sum\_nums a b lst)))))))

*Рекурсивная реализация.*

Устанавливаем начальные значения вспомогательных переменных:

(setf sum 0) - для подсчета искомой суммы чисел

(setf cnt 0) - для подсчета чисел (возможно не всех)

Функция, которая находит сумму чисел, удовлетворяющих условию:

(defun sum\_nums (a b lst sum)

(cond ((null lst) sum)

((and (numberp (car lst)) (evenp (car lst)) (>= (car lst) a) (<= (car lst) b)) (sum\_nums a b (cdr lst) (+ sum (car lst))))

((listp (car lst)) (sum\_nums a b (cdr lst) (sum\_nums a b (car lst) sum)))

(t (sum\_nums a b (cdr lst) sum))))

Эта же функция, но для юзера, чтобы убрать вспомогательную переменную sum из параметров.

(defun user\_sum\_nums (a b lst) (sum\_nums a b lst sum))

Функция, проверяющая, есть ли в списке числа. Если чисел нет, выводит 0, иначе - значение ненулевое.

(defun find\_nums (lst cnt)

(cond ((null lst) cnt)

((numberp (car lst)) (+ cnt 1))

((listp (car lst)) (find\_nums (cdr lst) (find\_nums (car lst) cnt)))

(t (find\_nums (cdr lst) cnt))))

(defun user\_find\_nums (lst) (find\_nums lst cnt))

(print (user\_find\_nums lst))

Эта же функция, но для юзера, чтобы убрать вспомогательную переменную cnt из параметров.

Функция, которая добавляет найденную сумму в качестве элемента списка на k-ую позицию.

(defun put\_sum (k lst sum)

(cond ((null lst) lst)

((= k 0) (cons sum lst))

(t (cons (car lst) (put\_sum (- k 1) (cdr lst) sum)))))

Основная функция, объединяющая работу всех предыдущих:

(defun work (a b lst k)

(cond ((= (user\_find\_nums lst) 0) (print "no numbers in the list!"))

(t (put\_sum k lst (user\_sum\_nums a b lst)))))

(print (work 0 10 lst 4))