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»

Рубежный контроль № 2

По дисциплине «Функциональное и логическое программирование»

Студент: Тимонин А. С.

Группа ИУ7-62б

Преподаватель Толпинская Н. Б.

Москва.
2020 г.

Практическая часть

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).

Через рекурсию

```
(defun getOneLstHelper (lst newlst)
     (cond ((null lst) newlst)
           ((atom lst) (cons lst newlst))
            (t (getOneLstHelper (car lst) (getOneLstHelper
(cdr lst) newlst) ))
     )
)
; getOneLstHelper – функция делающая из смешанного
; структурированного списка обычный смешанный список

(defun getOneLst (lst)
    (
        getOneLstHelper lst nil
    )
)
; getOneLst – функция обертка
; (getOneLst (list 1 2 3 (list 4 5 ) 6 (list 7 8 9 (list 10
11 12) ))

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



(defun goToN(lst n)
    (cond
        ( (or (< n 0) (> n (length lst))
              (null (car lst))) (list 0) )
```

```
        ( (= n 0) lst )
        ( t (goToN (cdr lst) (- n 1)))
    )
)
; goToN - функция, сдвигающая список к позиции n

(defun getSumHelper (lst sum)
    (cond
        ( (null (car lst)) sum )
        ( t (getSumHelper (cdr lst)
            (if (and (numberp (car lst)) (not (oddp (car lst))) )
                (+ sum (car lst))
                (+ sum 0)
            )
        ) )
    )
)
; getSumHelper - функция помошник, считающая
; сумму четных элементов, являющихся числом

(defun getSum(lst)
    (getSumHelper lst 0)
)
; getSum - функция-обертка, запускающая сдвинутый список
; на позицию a, и передающая этот параметр в функцию
помошник
; с параметром sum, равным 0



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

(defun getListABhelper (lst b)
    (cons
        (if (not (null lst))
            (car lst)
        )

        (cond
            ( (< b 0) nil)
            ((or (= b 0) (= (length (cdr lst)) 1)) (cons (cadr lst) nil))
            (t (getListABhelper (cdr lst) (- b 1)))
        )
    )
)
; getListABhelper - функция-обертка, формирующая список от
начала до позиции b


(defun getListAB (lst a b)
    (
```

```
        getListABhelper (goToN lst a) (- b (+ a 1))
    )
)
; getListAB – функция-обертка, формирующая список от
позиции a до позиции b



(defun insertHelper (lst pos num new_lst)
    (cond
        ( (null lst) new_lst )
        ( (= pos 0) (insert_element_helper (cdr lst) (- pos 1) num
(cons (car lst) (cons num new_lst))) )
        ( (insert_element_helper (cdr lst) (- pos 1) num (cons (car
lst) new_lst)) )
    )
)
; insertHelper – функция-помошник формирующая новый список
со вставленным элементом в позицию pos

(defun insertElement (lst num pos)
       (reverse (insertHelper lst pos num nil))
)
; insertElement – функция-обертка, вставляющая в любой
список число


(defun rk2 (lst a b k)
    (
       getOneLst (insertElement lst (getSum (getOneLst
(getListAB lst a b))) k)
    )
)

; rk2 – функция выполняющая задание к рк2 через рекурсию
Первым делом сдвигает список lst до позиций от a до b,
затем она разбивает все элементы списки внутри главного
списка, таким образом мы получаем в главном списке все
элементы. Далее мы вычисляем по этим элементам сумму четных
чисел. И в конце вставляем сумму четных чисел – sum на
позицию pos в нашем первоначальном списке


(rk2 (list 1 2 (list 1 2 3 4 (list 1 2 3 4 5) 5) 1 2 3) 2 2
2) -> (1 2 12 1 2 3 4 1 2 3 4 5 5 1 2 3)
```

Через функционалы

```
(defun getBetweenAB (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) (getBetweenAB x a b) )
        ) ) lst
    )
 )
; getBetweenAB - функция, которая возвращает список
элементы
; которого располагаются в границе от a до b


(defun getSum (lst a b)
       (reduce #'+ (getBetweenAB lst a b)))
; getSum - функция, возвращает сумму списка lst


 (defun isNum (lst)
    (mapcan #'(lambda (x)
        (cond
            ( (numberp x) (cons t nil) )
            ( (listp x) (isNum x) )
            ( (cons nil nil) )
        )
    )lst)
)
; isNum - функция возвращающая список из t и nil, t -
число, nil - не число

 (defun checkNumbers (lst)
       (and lst (reduce #'(lambda (a b)
            (or a b) ) (isNum lst)
                )
        )
)
; checkNumbers - функция проверяюшая наличие чисел в списке


 (defun rk2 (lst a b)
    (cond
         ( (checkNumbers lst) (nconc lst (cons (getSum lst
a b) nil)) )
```

```
          ( (princ "No num in list") )
    )
)
; rk2 – функция определяет есть ли в списке числа,
; если они есть, тогда ф-ция вычисляет сумму четных
; чисел в интервале от a до b и вставляет сумму в конец
; списка

(rk2 (list 1 2 (list 1 2 3 4 (list 1 2 3 4 5) 5) 1 2 3) 2 2
2) -> (1 2 12 1 2 3 4 1 2 3 4 5 5 1 2 3)
```