1 Лабораторная работа №6

Смирнов Пётр, ИУ7-65Б

1.1 Номер 1

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

```
(defun my-reverse(lst res)
    (cond ((null lst)
                    res)
          (t (my-reverse (cdr lst)
                         (cons (car lst)
                               res)))))

(my-reverse '(1 2 3 4 5) ())    ; (5 4 3 2 1)
(my-reverse '(1 (2 3) a 5) ()) ; (5 A (2 3) 1)
(my-reverse () ())              ; NIL
(my-reverse '(1) ())            ; (1)
```

1.2 Номер 2

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

```
(defun my(lst)
    (cond
          ((null lst)
            lst)
          ((and (listp (car lst))
                (car lst))
            (car lst))
          (t
            (my (cdr lst)))))

(my '(1 (2) (3) 4 5)) ; (2)
(my '(1 () (3) 4 5))  ; (3)
(my ())               ; NIL
(my '(((1) 2)))       ; ((1) 2)
```

1.3 Номер 3

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

а) все элементы списка – числа;

```
(defun my(lst num)
    (cond
          ((null lst))
          (t (rplaca lst
                     (* num (car lst)))
             (my (cdr lst)
                 num))))

(setf l1 '(1 2 3 4 5))
(my l1 -3)
(print l1)     ; (-3 -6 -9 -12 -15)
```

б) элементы списка – любые объекты.

```
(defun my(lst num)
    (cond
          ((null lst))
          ((cond
                ((numberp (car lst))
                   (rplaca lst
                           (* num (car lst))))
                ((listp (car lst))
                   (my (car lst) num))
                (t))

             (my (cdr lst) num))))

(setf l1 '(nil (2) a (4 5)))
(my l1 3)
(print l1)  ; (NIL (6) A (12 15))
```

1.4 Номер 4

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. (+ 2 балла)

```
(defun insert_help(x lst)
    (cond
          ((null lst)
              (list x))
          ((<= x (car lst))
              (cons x lst))
          (t  (cons (car lst)
                    (insert_help x (cdr lst))))))

(defun select-between(lst1 lst2 a b)
    (cond
          ((null lst1)
                  lst2)
          ((or (<= a (car lst1) b)
               (>= a (car lst1) b))
              (select-between (cdr lst1)
                              (insert_help (car lst1) lst2)
                              a b))
          (t      (select-between (cdr lst1)
                              lst2 a b))))


(select-between '(17 222 3 88 4 -5) () -100 100)  ;(-5 3 4 17 88)
(select-between '(17 222 3 88 4 -5) () 100 -100)  ;(-5 3 4 17 88)
(select-between '(1 1 1) () -100 100)             ;(1 1 1)
(select-between '(1 2) () -100 100)               ;(1 2)
(select-between '(2 1) () -100 100)               ;(1 2)
(select-between '(2 1 1) () 100 -100)             ;(1 1 2)
(select-between '(1) () 100 100)                  ;NIL
(select-between '(1) () -100 100)                 ;(1)
(select-between '(-1 -2 -3 -4 -5) () -100 100)    ;(-5 -4 -3 -2
   -1)
(select-between '() () -100 100)                  ;NIL
```

1.5 Номер 5

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

а) одноуровнего смешанного;

```
(defun rec-add(lst)
    (cond
          ((not lst) 0)
          ((numberp (car lst))
                (+ (car lst)
                   (rec-add (cdr lst))))
          (t    (rec-add (cdr lst)))))

(rec-add '(a 2 a 4 d)) ; 6
(rec-add '(1 2 3 4 5)) ; 15
(rec-add '(a b c d e)) ; 0
(rec-add ())           ; 0
```

б) структурированного.

```
(defun rec-add(lst)
    (cond
          ((numberp lst) lst)
          ((atom lst)    0)
          (t    (+ (rec-add (car lst))
                   (rec-add (cdr lst))))))

(rec-add '(1 2 (3) 4 5))       ; 15
(rec-add '(1))                 ; 1
(rec-add '(a b c d 3))         ; 3
(rec-add '(a b c d (3)))       ; 3
(rec-add '(1 2 (3 4 a b c) d)) ; 10
(rec-add '())                  ; 0
```

1.6 Номер 6

Написать рекурсивную версию с именем recnth функции nth.

```
(defun recnth(n lst)
    (cond ((not lst)  nil)
          ((minusp n) nil)
          ((eql n 0)  (car lst))
          (t (recnth (- n 1)
                     (cdr lst)))))

(recnth 100000000000 '(1 2 3 4 5)) ; NIL
(recnth 0 '(1 2 3 4 5))            ; 1
(recnth 4 '(1 2 3 4 5))            ; 5
(recnth -1 '(1 2 3 4 5))           ; NIL
```

1.7 Номер 7

Написать рекурсивную функцию allodd, которая возвращает t, когда все элементы списка нечётные.

```
(defun allodd(lst)
    (cond
          ((not (numberp (car lst)))
             nil)
          ((cdr lst)
             (and (oddp (car lst))
                  (allodd (cdr lst))))
          (t (oddp (car lst)))))

(allodd '(1 3 5 7))    ; T
(allodd '(2 4 6 8))    ; NIL
(allodd '(1 2 3 4))    ; NIL
(allodd ())            ; NIL
(allodd '(1))          ; T
(allodd '(1 a 2 b))    ; NIL
(allodd '(1 2 (3 4))) ; NIL
```

1.8 Номер 8

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

```
(defun first_odd(lst)
    (cond
          ((and (numberp lst)
                (oddp lst))
                lst)
          ((atom lst)
                nil)
          (t    (or (first_odd (car lst))
                    (first_odd (cdr lst))))))

(first_odd '(1 2 3 4 5))          ; 1
(first_odd '(2 3 4 5))            ; 3
(first_odd '())                   ; NIL
(first_odd '(a b c 4 6 1))        ; 1
(first_odd '((1 2) 3 4))          ; 1
(first_odd '(6 a 2 b (3) (4)))    ; 3
(first_odd '(2 4 6 8))            ; NIL
(first_odd '((2) ((3) 17)))       ; 3
```

1.9 Номер 9

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 kwad(lst)
    (cond
        ((not lst)
            lst)
        (t
            (cons (* (car lst)
                     (car lst))
                  (kwad (cdr lst))))))


(kwad '(1 2 3 4 5))    ; (1 4 9 16 25)
(kwad '(1))            ; (1)
(kwad ())              ; NIL
(kwad '(5 3 1))        ; (25 9 1)
```

1.10 Номер 10

Преобразовать структурированный список в одноуровневый.

```
(defun level (lst)
    (cond
        ((null lst)
            nil)
        ((atom (car lst))
            (cons (car lst)
                  (level (cdr lst))))
        (t  (nconc (level (car lst))
                   (level (cdr lst))))))

(level '(1 2 3 4 5))                ; (1 2 3 4 5)
(level ())                           ; NIL
(level '(1))                         ; (1)
(level '((1 2) (3 4)))               ; (1 2 3 4)
(level '((1) (2) ((3 4) (5)) 6 7)) ; (1 2 3 4 5 6 7)
(level '(1 2 3 ((4)) 5 (6 7)))      ; (1 2 3 4 5 6 7)
```