1 Лабораторная работа №5

Смирнов Пётр, ИУ7-65Б

1.1 Номер 1

Напишите функцию, которая уменьшает на 10 все числа из списка-аргумента этой функции, проходя по верхнему уровню списковых ячеек. (список смешанный структурированный)

```
(defun fc(e)
    (cond ((numberp e) (- e 10))
          (t e)))

(defun my(arr) (mapcar #'fc arr))

(my '(a 2 3 4 5))       ;(A -8 -7 -6 -5)
(my '(a (c d) 3 4 5)) ;(A (C D) -7 -6 -5)
(my '())                ;NIL
(my '((2)))             ;((2))
(my '(10))              ;(0)
```

1.2 Номер 2

Написать функцию, которая получает как аргумент список чисел, а возвращает список квадратов этих чисел в том же порядке.

```
(defun my_square (x) (* x x))
(defun my(arr) (mapcar #'my_square arr))

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

1.3 Номер 3

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

а) все элементы списка – числа;

б) элементы списка – любые объекты.

```
(defun my(arr number)
    (mapcar #'(lambda (e) (* e number)) arr))

(my '(1 2 3 4 5) 3) ;(3 6 9 12 15)
(my '() 3)          ; NIL
```

```
(defun my(arr number)
    (mapcar #'(lambda (e)
                (cond ((numberp e) (* e number))
                      (t e))) arr))

(my '(1 (2) 3 4 5) 3) ; (3 (2) 9 12 15)
(my '() 3)            ; NIL
```

1.4 Номер 4

Написать функцию, которая по своему списку-аргументу lst определяет, является ли он палиндромом для одноуровнего смешанного списка.

```
; отсекает последний элемент
(defun var2(arr)
    (cond ((not (cdr arr)) (rplaca arr nil))
          (t (rplacd (nthcdr (- (length arr) 2) arr) nil) arr)))

; проверяет равенство крайних
(defun predic(l)
    (cond ((not (cdr l)))
          (t (cond ((equal (car (last l)) (car l)) (var2 l) t)
                   (t (var2 l) nil)))))

; проверяет что итоговый список состоит из t
(defun check(lst)
    (every #'(lambda(x) x)
           (maplist #'predic lst)))

(check '(1 a 3 a 1)) ; T
(check '(1 2 2 1))   ; T
(check '(1 2 3 1 1)) ; NIL
(check '(1))         ; T
(check '())          ; T
```

1.5 Номер 5

Написать предикат set-equal, который возвращает t, если два его множества-аргумента (одноуровневые списки) содержат одни и те же элементы, порядок которых не имеет значения.

```
(defun create_predicat(l1 l2)
    (mapcar #'(lambda (e) (cond ((member e l1) t))) l2))

(defun check(lst)
    (every #'(lambda(x) x) lst))

(defun set-equal(l1 l2)
    (and (check (create_predicat l1 l2))
         (check (create_predicat l2 l1))))


(set-equal '(1 2 3) '(1 2 3 4)) ; NIL
(set-equal '(1 2 3) '(1 2 3))   ; T
(set-equal '(9 8 6 4) '(1 2 3)) ; NIL
(set-equal '(3 2 1) '(1 2 3))   ; T
(set-equal '() '())             ; T
(set-equal '(1) '(1))           ; T
```

1.6 Номер 6

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

```
(defun forreduce(a b)
    (cond ((numberp a)
            (cond ((< a b) (cons a (cons b nil)))
                  (t (cons b (cons a nil)))))
          (t (append (remove-if #'(lambda(x) (>= x b)) a)
                     (cons b nil)
                     (remove-if #'(lambda(x) (< x b)) a)))))

(defun select-between(lst a b)
    (let ((no-trash (remove-if-not #'(lambda(x) (or (<= a x b)
                                                    (>= a x b)))
                                   lst)))
        (cond ((not (cdr no-trash)) no-trash)
              (t (reduce #'forreduce no-trash)))))

(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
```

1.7 Номер 7

Написать функцию, вычисляющую декартово произведение двух своих списков-аргументов.

```
(defun decart(lstX lstY)
    (mapcar #'(lambda(x) (mapcar #'(lambda(y) (cons x y)) lstY))
            lstX))

(decart '(1 2 3) '(a b c)) ; (((1 . A) (1 . B) (1 . C)) ((2 . A)
   (2 . B) (2 . C)) ((3 . A) (3 . B) (3 . C)))
(decart '(1 2 3) '(a b))   ; (((1 . A) (1 . B)) ((2 . A) (2 . B))
   ((3 . A) (3 . B)))
```

1.8 Номер 8

Почему так реализовано reduce, в чём причина?

```
(reduce #'+ ()) ; 0
(reduce #'* ()) ; 1

(+) ; 0
(*) ; 1
```

Если список, подаваемый на вход пуст, то функция, указанная вторым аргументом reduce вызывается без аргументов. +, вызываемый без аргументов возвращает 0. *, вызываемая без аргументов, возвращает 1.

1.9 Номер 9

Пусть list-of-list список, состоящий из списков. Написать функцию, которая вычисляет сумму длин всех элементов list-of-list (количество атомов).

```
(defun func(lst) (apply #'+ (mapcar #'(lambda(x) (length x)) lst))
   )

(func '((1) (2) (3 4) (7 8 0))) ; 7
(func '())                      ; 0
(func '(()))                    ; 0
(func '((1)))                   ; 1
```