1. Написать функцию, которая по своему списку-аргументу lst определяет является ли он палиндромом (то есть равны ли lst и (reverse lst)).

```
(defun pal (lst rev len)
   (cond   ((<= len 0) t)
         (   (and    (equal (car lst) (car rev))
                (pal (cdr lst) (cdr rev) (- len 2))
            )
            t
         )
   )
)
```

4. Напишите функцию swap-first-last, которая переставляет в списке-аргументе первый и последний элементы.

```
 (defun swap-first-last (lst)
    (append (last lst) (cdr (butlast lst)) (cons (car lst) nil))
 )
```

5. Напишите функцию swap-two-element, которая переставляет в списке-аргументе два указанных своими порядковыми номерами элемента в этом списке.

```
(defun swap-two-element (lst f s)
   (let    ((temp (nth f lst)))
         (setf (nth f lst) (nth s lst ))
         (setf (nth s lst) temp))
   lst
)
```

6. Напишите две функции, swap-to-left и swap-to-right, которые производят круговую перестановку в списке-аргументе влево и вправо, соответственно.

```
(defun swap-to-left (lst)
   (append (cdr lst)
         (cons (first lst) nil)
   )
)
(defun swap-to-right (lst)
   (append (last lst)
         (butlast lst)
   )
)
```

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

а) все элеметны списка - числа,

```
(defun multiply-all (lst mul)
   (mapcar #'(lambda (x) (* x mul))
         lst
   )
)
```

б) элементы списка - любые объекты.

```
(defun multiply-all (lst mul)
   (mapcar #'(lambda (x)
            (cond   ((numbperp x) (* x mul))
                  ((listp x) (multiply-all x mul))
                  (t x)
            )
         )
         lst
   )
)
```

8. 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 списка чисел).

```
(defun select-between (lst left right)
   (remove-if  #'(lambda (x) (or (< x left) (> x right)))
            lst)
)
```