Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 8

Дисциплина	Функциональное и логическое программирование
Студент	Сиденко А.Г.
Группа	ИУ7-63Б
Преподаватель	Толпинская Н.Б., Строгано Ю.В.

Москва, 2020 г.

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

```
(defun palindrom (lst)
  (equalp lst (reverse lst))
)
```

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

   На вход подается список.

   Функция last_elem ищет последний элемент в списке.

   Функция centr возвращает список без последнего элемента.

   Функция swap_first_last объединяет последний, середину и первый элемент исходного списка. Символ ‘ блокирует вычисления, но эту блокировку можно прервать с помощью другого символа , .

```
(defun last_elem (lst)
  (if (NULL (cadr lst))
    (car lst) (last_elem (cdr lst))
  )
)

(defun centr (lst)
  (if (NULL (cadr lst))
    ()
    (cons (car lst)(centr (cdr lst)))
  )
)

(defun swap_first_last (lst)
  `(,(last_elem lst) ,@(centr (cdr lst)) ,(car lst))
)
```

3. **Напишите функцию swap-two-ellement, которая переставляет в списке-аргументе два указанных своими порядковыми номерами элемента в этом списке.**

   На вход подается список и 2 числа.

   Используется та же идея объединения списков, что и в предыдущем пункте, но сначала проверяем чтобы первый порядковый номер был меньше второго, в противном случае вызывается функция с переставленными аргументами.

Объединяются списки с 0 до i, j элемент, с i+1 до j, i элемент и с j+1 до конца.

```
(defun swap_two_ellement (lst i j)
  (cond ((= i j) lst)
    ((> i j) (swap_two_ellement lst j i))
    (t
     `(,@(subseq lst 0 i),(nth j lst),@(subseq lst(+ i 1)j)
      ,(nth i lst) ,@(subseq lst (+ j 1))))
  )
)
```

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

   На вход подается список и число.

   Та же идея объединения списков, в зависимости от перестановки влево или вправо, объединяются разные списки. Если количество позиций в перестановке больше длины, берется остаток от деления функция rem.

```
(defun swap_to_left (lst k)
  (cond ((= k 0) lst)
        ((> k (length lst))
         (swap_to_left lst (rem k (length lst))))
        (t
         `(,@(subseq lst k) ,@(subseq lst 0 k)))
   )
)

(defun swap_to_right (lst k)
 (cond ((= k 0) lst)
       ((> k (length lst))
        (swap_to_right lst (rem k (length lst))))
       (t
        `(,@(subseq lst (- (length lst) k))
        ,@(subseq lst 0 (- (length lst) k))))
 )
)
```

5. **Напишите функцию, которая умножает на заданное число-аргумент все числа из заданного списка-аргумента, когда все элементы списка — числа или элементы списка – любые объекты.**

   На вход подается список и число.

С помощью функционала mapcar, производится работа с каждым аргументом списка: проверяется на число (если необходимо), выполняется умножение на заданный аргумент.

```
(defun multiplication_numbers (lst k)
  (mapcar #'(lambda (x) (* x k)) lst)
)

(defun multiplication_all (lst k)
  (mapcar #'(lambda (x) (if (numberp x) (* x k) x)) lst)
)
```

6. **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).**

На вход подается 2 числа и список.

Используется функционал reduce, определяется функция found, для проверки вхождения в диапазон. Данная функция будет применяться каскадным образом (к первым двум, затем к результату и следующему и так далее). Следовательно, нужно проверить является ли первый аргумент числом (в первый раз), если так то проверяются оба, нет один. С помощью функции append создается список всех подходящих значений.

```
(defun found_between (a b lst)
  (defun found (lst1 lst2)
    (if (numberp lst1)
      (append(if(and(< a lst1)(< lst1 b))(list lst1) Nil)
             (if(and(< a lst2)(< lst2 b))(list lst2) Nil))
      (append lst1
             (if(and(< a lst2)(< lst2 b))(list lst2) Nil))
    )
  )
  (reduce #'found lst)
)
```