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»

Лабораторная работа №5

По предмету: «Функциональное и логическое программирование»

Студент: Гасанзаде М.А.,
Группа: ИУ7-66Б

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) (equal lst (reverse lst)) )
```

Или

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

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

```
(defun set-equal (x y) (and (subsetp x y)  (subsetp y x)))
```

3. Напишите необходимые функции, которые обрабатывают таблицу из точечных пар (страна.столица), и возвращают по стране - столицу, а по столице - страну.

```
(defun check (pair val) ;возвращает противоположный элемент из точечной пары если
проходит совпадение, иначе - нил
    (cond   ((equal (car pair) val) (cdr pair))
            ((equal (cdr pair) val) (car pair))
    )
)
(defun generate-check (val) ;возвращает функцию, которая будет сравнивать свой
параметр с этим самым валом
    (lambda (pair) (check pair val))
)

;создаём функцию для сверки с интересующим нас валом; проходим ею по всей таблице;
выдергиваем первый ненулевой результат
(defun find-in-table (base val)
    (find-if    #'(lambda (x) (not (eq x Nil)))
                (mapcar (generate-check val) base)
    )
)
```

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-ellement, 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 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)
    )
)
```

```
1)(defun left (lst)
    (reverse (swap-to-right (reverse lst)))
)

2) length-1 раз вызвать swap-to-right
```

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 (+ 2 балла)).

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