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

О т ч е т

по лабораторной работе № 8

Дисциплина: «Функциональное и логическое программирование»

Выполнила: Овчинникова А.П.

Группа: ИУ7-65Б

Преподаватель: Толпинская Н.Б.

Строганов Ю.В.

Москва, 2020

Теоретическая часть.

Функция *(equalp object1 object2)* возвращает истину, если *object1* и *object2* равны с точки зрения equal, char-equal или =; либо являются cons-ячейками, car и cdr которых эквивалентны с точки зрения *equalp*; либо являются массивами одинаковой длины, элементы которых эквивалентны с точки зрения *equalp*; либо являются структурами одного типа, элементы которых равны с точки зрения *equalp*; либо являются хеш-таблицами с одинаковыми тестовыми функциями и количеством элементов, ключи которых связаны со значениями, равными для двух таблиц с точки зрения *equalp*. Вызов с циклическими аргументами может не завершиться.

Функция *(null object)* возвращает истину, если объект *object* является nil.

Функция *(rplaca cons object)* эквивалентна *(setf (car cons) object)*, но возвращает cons.

Функция *(nthcdr N lst)* эквивалентна n-кратному последовательному применению cdr к lst.

Функция *(not object)* возвращает истину, если объект *object* имеет значение nil.

Функция *(remove el lst)* возвращает последовательность, похожую на lst, но без всех элементов, совпадающих с el.

Практическая часть.

Задание 1.

Написать функцию, которая по своему списку-аргументу определяет, является ли он палиндромом.

```
(defun is_palindrome (lst)
   (equalp lst (reverse lst))
)
```

Задание 4.

Написать функцию swap-first-last, которая приставляет в списке-аргументе первый и последний элементы.

```
(defun swap-first-last (lst)
```

```
    (and (setf tmp (car lst))
        (setf (car lst) (car(last lst)))
        (setf (car (last lst)) tmp)
        lst
    )
)
(defun swap-first-last2 (lst)
    (cond ((null lst) lst)
          ((not (null lst))
            (and
                (setf tmp  (car lst))
                (rplaca lst (car (last lst)))
                (setf (car (last lst)) tmp)
                lst
            )
          )
    ))
(defun swap_first_last3 (lst)
    (if (null (cdr lst))
        lst
        (append
            (cons
                (car (last lst))
                (reverse (cdr (reverse (cdr lst))))
            )
        (cons (car lst) Nil)
        )
    ))
(defun swap_first_last4 (lst)
    (append (last lst) (cdr (butlast lst 1)) (cons (car lst) Nil))
```

```
)
```

Задание 5.

Написать функцию swap-two-elements, которая переставляет в списке-аргументе два указанных своими порядковыми номерами элемента.

```
(defun swap-two-element (lst el1 el2)
   (setf tmp (nth el1 lst))
   (rplaca (nthcdr el1 lst) (nth el2 lst))
   (rplaca (nthcdr el2 lst) tmp)
   lst
)
(defun swap-two-element2 (lst el1 el2)
   (cond ( (null lst) lst )
         ( (not (plusp (+ el1 1))) lst )
         ( (not (plusp (+ el2 1))) lst )
         ( (>= el1 (list-length lst)) lst )
         ( (>= el2 (list-length lst)) lst )
         (
           (setf tmp (nth el1 lst))
           (rplaca (nthcdr el1 lst) (nth el2 lst))
           (rplaca (nthcdr el2 lst) tmp)
           lst
         )
   )
)
```

Задание 6.

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

```
 (defun shl2 (lst)
   (append (cdr lst) (cons (car lst) Nil))
)
```

```
(defun shl (lst)
   (cons (cdr lst) (cons (car lst) Nil))
)
(defun swap-to-left (lst k)
   (loop for x from 0 to (- k 1)
      do (setf lst (shl lst))
   )
   lst
)
(defun shr2 (lst)
   (append (last lst) (reverse (cdr (reverse lst))))
)
(defun shr (lst)
   (cons (car (last lst)) (reverse (cdr (reverse lst))))
)
(defun swap-to-right (lst k)
   (loop for x from 0 to (- k 1)
      do (setf lst (shr lst))
   )
   lst
)
(defun swap-to-left3 (lst k)
   (do
      ((x 1 (+ x 1)))
      ((> x k))
      (setf lst (shl2 lst))
   )lst)
```

Задание 7.

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

- все элементы списка – числа;
- элементы списка – любые объекты.

```
(defun mult (lst num)
   (defun m (n)
      (* n num)
   )
   (mapcar #'m lst)
)
(defun mult2 (lst num)
   (defun m (n)
      (cond ( (numberp n) (* n num) )
            (n)
      )
   )
   (mapcar #'m lst)
)
```

Задание 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.

```
(defun select-between (lst min max)
   (defun check (n)
      (cond ( ( and (<= n max) (>= n min) ) n)
            (Nil)
      )
   )
   (remove Nil (mapcar #'check lst))
)
(defun select-between2 (lst min max)
   (mapcan #'(lambda (n)
```

```
            (cond ( ( and (<= n max) (>= n min) ) (list n))
                (Nil)
            )
        )
        lst
    )
)
```