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 № 8

Студент Лучина Е.Д

Группа ИУ7-61Б

Преподаватель Толпинская Н.Б.

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 my_reverse (lst) (cond ((null lst) Nil) ((eq (cdr lst) Nil) (cons (car lst) Nil)) (t (append (my_reverse (cdr lst)) (cons (car lst) Nil))))) (defun func (lst) (equal lst (my_reverse lst)))	(func '()) (func '(1)) (func '(1 2)) (func '(1 2 1)) (func '(1 2 3 3 2 1)) (func '(1 4 2 1 4 5))	T T Nil T T Nil

Функция my_reverse оборачивает список, не разрушая его структуру. Func принимает в качестве аргумента список, возвращает результат сравнения его с перевернутым списком, в случае равенства списков (то есть список является палиндромом) - T, иначе Nil.

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)))	(swap-first-last '())) (print(swap-first-last '(1)) (swap-first-last '(1 2)) (swap-first-last '(1 2 3)) (swap-first-last '(1 2 3 4 5 6))	(NIL) (1 1) (2 1) (3 2 1) (6 2 3 4 5 1)

Функция swap-first-last использует встроенную функцию append, для конкатенации в нужном порядке следующих списков: последний элемента, хвост списка исключая последний элемент, точечную пару (первый элемент . Nil).

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

<table>
<tr><td rowspan="4">(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
)</td><td>(swap-two-element '(1 2 3 4 5) 0 0)</td><td>(1 2 3 4 5)</td></tr>
<tr><td>(swap-two-element '(1 2 3 4 5) 0 4)</td><td>(5 2 3 4 1)</td></tr>
<tr><td>(swap-two-element '(1 2 3 4 5) 3 1)</td><td>(1 4 3 2 5)</td></tr>
<tr><td colspan="2">(swap-two-element '(1 2 3 4 5) 3 5) -> ERROR
(SETF NTH): index 5 is too large for (1 2 3 NIL 5)</td></tr>
</table>

Используя возможность определения локальной переменной, с помощью let объявляем временный буфер temp, в который записывается значение элемента списка с индексом f. Далее элементу списка с индексом f присваивается значение элемента с индексом s, а элементу с индексом f значение из временного буфера temp. Возвращаем список. Для присваивания используется функция setf.

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

<table>
<tr><td colspan="2">(defun swap-to-left (lst k)
  (cond
  ((null lst) Nil)
  ((eq (cdr lst) Nil) lst)
  ((eq k 0) lst)
  (t (swap-to-left
  (append (cdr lst) (cons (first
  lst) nil)) (- k 1)))
)
)</td><td colspan="2">(defun swap-to-right (lst k)
  (cond
  ((null lst) Nil)
  ((eq (cdr lst) Nil) lst)
  ((eq k 0) lst)
  (t (swap-to-right
  (append (last lst) (butlast
  lst)) (- k 1)))
)
)</td></tr>
<tr><td>(swap-to-left '(1 2 3 4 5) 0)</td><td>(1 2 3 4 5)</td><td>(swap-to-right '(1 2 3 4 5) 0)</td><td>(1 2 3 4 5)</td></tr>
<tr><td>(swap-to-left '(1 2 3 4 5) 1)</td><td>(2 3 4 5 1)</td><td>(swap-to-right '(1 2 3 4 5) 1)</td><td>(5 1 2 3 4)</td></tr>
<tr><td>(swap-to-left '(1 2 3 4 5) 3)</td><td>(4 5 1 2 3)</td><td>(swap-to-right '(1 2 3 4 5) 3)</td><td>(3 4 5 1 2)</td></tr>
</table>

(swap-to-left '() 3)	Nil	(swap-to-right '(1 2 3 4 5) 3)	Nil
(swap-to-left '(1) 3)	(1)	(swap-to-right '(1 2 3 4 5) 3)	(1)

Если список пуст - возвращает Nil, если список имеет один элемент - возвращает этот же список, если k = 0, возвращает список. Иначе запускается рекурсия, которая передает в функцию модифицированный список и документированную k. В случае сдвига влево - модифицированный список представляет собой хвост исходного, дополненного точечной парой (первый элемент . Nil). В случае сдвига вправо - последний элемент, дополненный исходным списком без последнего элемента.

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

а) когда все элементы списка - числа

функционал	Дополняемая рекурсия
(defun func (lst a) (mapcar (lambda (x) (* a x)) lst))	(defun r_func (lst a) (cond ((null lst) nil) (t (cons (* a (car lst)) (r_func (cdr lst) a)))))

Пример работы: (func '(1 2 3) 2) -> (2 4 6)

б) когда элементы списка - любые объекты.

функционал	Дополняемая рекурсия
(defun func2 (lst a) (mapcar (lambda (x) (if (numberp x)	(defun r_func2 (lst a) (if (not (null lst)) (cons (if (numberp (car lst))

```
      (* a x)
      x
    )
  )
  lst
 )
)
```

```
        (* a (car lst))
        (car lst)
      )
      (r_func2 (cdr lst) a)
    )
  )
)
```

Пример работы: (func2 '(1 d (3 4) 3) 2) -> (1 d (3 4) 6)

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 a b)
   (cond
      ((null lst) Nil)
      ((and
        (numberp (car lst))
        (< (car lst) b)
        (> (car lst) a))
        (cons (car lst) (select-between (cdr lst) a b)))
      (t (select-between (cdr lst) a b))
   )
)

(defun sorted-select-between (lst a b)
   (sort (select-between lst a b) #'<)
)
```

Если список пуст вернет Nil. Если первый элемент является числом, которое меньше b и больше a, добавляем его в результирующий список с помощью cons. Вернем точечную пару (этот элемент , обработанный хвост списка). Хвост списка обрабатывается рекурсивно. Если первый элемент не удовлетворяет условиям, его пропускаем, вернем обработанный хвост. Сортировка осуществляется функцией sort, укажем #’< для сортировки по возрастанию.

Пример работы: (sorted-select-between '(0 5 -2 -5 3 1 2 -7) -2 4) -> (0 1 2 3)