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»

Дисциплина: «Функциональное и логическое программирование»

Лабораторная работа №8

Студент: Левушкин И. К.
Группа: ИУ7-62Б
Преподаватели: Толпинская Н. Б.,
Строганов 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)).

Реализация задания

```
(defun polindrom (lst)
    (and lst
        (reduce (lambda (prev x) (and prev x))
            (mapcar #'equal lst (reverse lst)))
    )
)
```

Рис. 1: Функция, проверяющая список на полиндром

Назначение параметров функций

- Функционал mapcar возвращает список типа (nil t t nil nil)
- Функционал reduce применяет and для списка

Результаты работы

Выражение	Результат
'(1 2 3 4 5)	NIL
'(1 2 3 2 1)	T
'(1 2 3)	NIL
'(1)	T
'()	NIL
'((1 2) 4 (1 2))	NIL

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

Реализация задания

```
(defun swap-first-last(lst)
    (cond ((< (length lst) 2) lst)
        (t (cons (
                mapcon #'(lambda (el);поиск последнего элемента в списке
                    (cond
                        (
                            (null (cdr el)) (car el)
                        )
                    )
                ) lst
            )
            (
                nconc (
                    mapcon #'(lambda (el);берем все элементы начиная со второго кроме последнего
                        (cond
                            (
                                (cdr el) (cons (car el) nil)
                            )
                        )
                    )
                    (cdr lst)
                )
                (cons (car lst) nil);добавляем первый элемент в конец
            )
        ))
    )
)
```

Рис. 2: Функция, переставляющая в списке-аргументе первый и последний элементы

Назначение параметров функций

- Первый функционал mapcon ищет последний элемент в списке
- Второй функционал mapcon формирует список, начиная со второго и до последнего (не включительно)
- Если длина списка меньше 2, то условие cond возвращает список без изменений

Результаты работы

Выражение	Результат
’(1 2 3 4 5 6)	(6 2 3 4 5 1)
’(1 2 3 4 5 (1 2))	((1 2) 2 3 4 5 1)
’(1 2)	(2 1)
’(1)	(1)
’()	NIL

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

Реализация задания

```
(defun my_nthcdr (lst num);поиск списка, начинающегося с num в списке
    (let ((after (- (length lst) num )))
                (maplist #'(lambda (el)
                    (cond
                        (
                            (equal (length el) after) el
                        )
                    )
                ) lst)
    )
)


(defun reduce_or (lst); применяем or к списку
    (reduce #'(lambda (prev lst) (or prev lst)) lst)
)

(defun from_to_list (lst from to);поиск списка, начинающегося с from до to в списке(все включительно)
    (cond ((or (< from 0) (>= from (length lst)) (< to 0) (>= to (length lst))) nil)
        (t
            (let* ((to_position_in_new_list (- to from));позиция элемента с индексом to в списке без from
                    (from_list (reduce_or (my_nthcdr lst from))); список начиная с from
            )
            (reverse (reduce_or (my_nthcdr (reverse from_list) (- (length from_list) (+ to_position_in_new_list 1)))))
            )
        )
    )
)
```

Рис. 3: Вспомогательные функции для swap-two-element

```
(defun my_nth_list (pos lst);если не nil, оборачиваем в список, иначе просто nil
    (let ((answer (nth pos lst)))
        (if answer (list answer) nil)
    )
)

(defun swap-two-element(lst num1 num2)
    (cond ((equal num1 num2) lst)
            ((or (>= num1 (length lst)) (>= num2 (length lst))) lst)
            (t (let ((before (min num1 num2))
                    (after (max num1 num2))
                )
                (nconc
                    (from_to_list lst 0 (- before 1))
                    (my_nth_list after lst)
                    (from_to_list lst (+ before 1) (- after 1))
                    (my_nth_list before lst)
                    (from_to_list lst (+ after 1) (- (length lst) 1))
                )
            ))
    )
)
```

Рис. 4: Функция, переставляющая в списке-аргументе два указанных своими порядковыми номерами эемента в этом списке

Назначение параметров функций

- Параметр before - минимальное число из num1 и num2
- Параметр after - максимальное число из num1 и num2
- Далее формируем и конкатенируем следующие списки из списка lst: (от нуля и до before - 1); (after); (от before + 1 и до after - 1); (before); (от after + 1 и до конца списка)
- Функция my_nth_list возвращает элемент из списка на позиции pos, оборачивая его в список, если он не nil
- Фцункция from_to_list формирует список из списка-параметра, начиная с позиции from и до to (включительно)
- Функция reduce_or применяет or к списку
- Функция my_nthcdr возвращает хвост списка lst, начиная с позиции num

Результаты работы

Выражение	**Результат**
'(1 2 3 4 5 6 7 8) 1 4	(1 5 3 4 2 6 7 8)
'(1 2 3 4 5 6 7 8) 4 1	(1 5 3 4 2 6 7 8)
'(1 2 3 4 5 6 7 8) 0 7	(8 2 3 4 5 6 7 1)
'(1 2 3 4 5 6 7 8) 1 8	(1 2 3 4 5 6 7 8)
'(1 2 3 4 5 6 7 8) 1 10	(1 2 3 4 5 6 7 8)
'(1 2 3 4 5 6 7 8) 1 1	(1 2 3 4 5 6 7 8)
'(1 2) 1 1	(1 2)
'(1 2) 0 1	(2 1)
'(1) 0 0	(1)
'() 0 0	NIL

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

Реализация задания

```
(defun get_k_cdr_elem (lst k)
    (if (equal k 0) lst (get_k_cdr_elem (cdr lst) (- k 1)))
)


(defun create_circle_list (lst)
    (rplacd (get_k_cdr_elem lst (- (length lst) 1)) lst)
)
```

Рис. 5: Вспомогательные функции к фукциям swap-to-left и swap-to-right

```
(defun swap-to-right (lst k)
    (cond ((null lst) nil)
    (t (let* ((lst_length (length lst))
            (mod_k (mod k lst_length))
            (circle_list (create_circle_list lst))
            (new_first_elem (get_k_cdr_elem circle_list (- lst_length (- mod_k 1))))
            (last_elem (rplacd (get_k_cdr_elem circle_list (- lst_length mod_k)) nil))
    )
        new_first_elem
    ))
    )
)

(defun swap-to-left (lst k)
    (cond ((null lst) nil)
    (t(let* ((lst_length (length lst))
            (mod_k (mod k lst_length))
            (circle_list (create_circle_list lst))
            (new_first_elem (get_k_cdr_elem circle_list (+ mod_k 1)))
            (last_elem (rplacd (get_k_cdr_elem circle_list mod_k) nil))
    )
        new_first_elem
    ))
    )
)
```

Рис. 6: Функции, производящие круговую перестановку на k позиций в списке-аргументе влево и вправо, соответственно

Назначение параметров функций

- Переменная lst_length обозначает длину первоначального списка (до его изменения функцией rplacd)
- Переменная mod_k берет остаток от деления k на длину списка
- Переменная new_first_elem - новая голова списка
- Переменная last_elem - результат работы функции get_k_cdr_elem (последний элемент списка)
- Функция create_circle_list делает список lst кольцевым (переставляет указатель последнего элемента на голову списка и возвращает этот последний элемент)
- Функция get_k_cdr_elem идет по списку и возвращает указатель на его элемент через k позиций

- Функция swap_to_right аналогична функции swap_to_left за исключением того, что она сдвигает список на length(lst) - (k + 1) позиций вместо (k + 1)

Результаты работы

Выражение	**Результат swap_to_left**	**Результат swap_to_right**
'(1 2 3 4 5 6) 3	(4 5 6 1 2 3)	(4 5 6 1 2 3)
'(1 2 3 4 5 6) 6	(1 2 3 4 5 6)	(1 2 3 4 5 6)
'(1 2 3 4 5 6) 9	(4 5 6 1 2 3)	(4 5 6 1 2 3)
'(1 2 3 4 5 6) 0	(1 2 3 4 5 6)	(1 2 3 4 5 6)
'(1 2 3 4 5 6) 2	(3 4 5 6 1 2)	(5 6 1 2 3 4)
'(1) 0	(1)	(1)
'(1) 3	(1)	(1)
'(1) 0	(1)	(1)
'() 0	NIL	NIL

7. Напишите функцию, которая умножает на заданное число-аргумент все числа из заданного списка-аргумента, когда а) все элементы списка — числа, б) элементы списка – любые объекты.

Реализация задания

```
(defun multiply_numbers (lst number)
    (mapcar (lambda (x) (* x number)) lst)
)

(defun multiply (lst number)
    (mapcar (lambda (obj)
                (cond ((numberp obj) (* obj number))
                      ((listp obj) (multiply obj number))
                      (t obj)
                )
            )
    lst)
)
```

Рис. 7: Функции multiply_numbers (элементы списка - числа) и multiply (элементы списка - любые объекты), умножающие число-аргумент на числа списка-аргумента

Назначение параметров функций

- Функция multiply_numbers выполняет пункт а из задания
- Функция multiply выполняет пункт б из задания, используя проверку на число (функция numberp) и проверку на список (функция listp)

Результаты работы

Выражение	Результат multiply_numbers	Результат multiply
'(1 2 3 4 5 6) 3	(3 6 9 12 15 18)	(3 6 9 12 15 18)
'(1 2 3 4 5 6) 0	(0 0 0 0 0 0)	(0 0 0 0 0 0)
'(1 2 3 4 5 6) 1	(1 2 3 4 5 6)	(1 2 3 4 5 6)
'() 5	NIL	NIL
'(1 2 3 (1 2) 4 5) 4	—	(4 8 12 (4 8) 16 20)
'(1 2 3 (1 2 (a 3)) 4 5) 6	—	(6 12 18 (6 12 (A 18)) 24 30)

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)).

Реализация задания

```
(defun select-between (lst from to)
    (let ((min_el (min from to))
          (max_el (max from to))
    )
        (apply #'nconc (;конкатенирую все подсписки списка
            mapcar (lambda (el) (if (and (< el max_el) (> el min_el)) (cons el nil) nil)) lst
        ))
    )
)
```

Рис. 8: Функция, выбирающая из списка-аргумента элементы меньшие to и большие from

```
(defun sort_insert (head before lst elem)
    (cond
        ((null head) (cons elem head))
        ((null lst) (let ((change_list (rplacd before (cons elem nil))))
                          head
                    )
        )
        (t (if (< elem (car lst)) (let* (  (list_elem (cons elem nil))
                                           (next_list (rplacd list_elem lst));вставляем элемент между before и lst
                                           (before_list (if (null before) nil (rplacd before list_elem)))
                                           )
                                           (if (null before) next_list head)
                                        )
                (sort_insert head lst (cdr lst) elem)
            )
        )
    )
)
```

Рис. 9: Вспомогательная функция сортировки для функции select-between

```
(defun check (result lst from to)
    (if (null lst) result
        (check (if (and (< (car lst) to) (> (car lst) from)) (
            sort_insert result nil result (car lst)
        ) result) (cdr lst) from to)
    )
)

;отсортированный вариант
(defun select-between (lst from to)
    (let ((min_el (min from to))
            (max_el (max from to))
    )
        (let ((result nil))
            (check result lst min_el max_el)
        )
    )
)
```

Рис. 10: Функция, выбирающая из списка-аргумента элементы меньшие to и большие from и сортирующая их по возрастанию

Назначение параметров функций

- Переменные min_el, max_el - минимальные и максимальные границы из from и to соответственно
- Идея фнкции sort_insert аналогична сортировке вставками: вставляет число-аргумент перед ближайшим большим элементом списка-аргумента в этот список
- Переменная head - голова списка
- Переменная before - предыдущий элемент списка (изначально nil)
- Переменная change_list - результат работы функции rplacd, которая перебрасывает указатель элемента списка before на elem
- Переменная next_list - результат работы функции rplacd, вставляющей elem в голову списка lst
- Переменная before_list - результат работы функции rplacd, которая перебрасывает указатель элемента списка before на elem, если before не null

- Функция check - хвостовая рекурсивная функция, аналогичная select-between без сортировки

Результаты работы

Выражение	**Результат select-between**	**Результат сорт. select-between**
'(1 9 7 3 4 6 8 2 5) 3 7	(4 6 5)	(4 5 6)
'(1 9 7 3 4 6 8 2 5) 0 10	(1 9 7 3 4 6 8 2 5)	(1 2 3 4 5 6 7 8 9)
'(1 9 7 3 4 6 8 2 5) 7 3	(4 6 5)	(4 5 6)
'(1 9 7 3 4 6 8 2 5) 3 3	NIL	NIL
'(1 9 7 3 4 6 8 2 5) 3.3 4.3	(4)	(4)
'(1 9) 3.3 4.3	NIL	NIL
'(1 9) 10.3 9.3	NIL	NIL
'(1) 0.3 1.3	(1)	(1)
'() 0.3 1.3	NIL	NIL