



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

Тема Функции более высокого порядка

Студент Сушина А.Д.

Группа ИУ7-616

Оценка (баллы) _____

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 reverse-list (list)
  (reduce
    (lambda (tmp result)
      (append result (list tmp)))
    list :from-end t :initial-value nil
  )
)
```

```
(defun is-palindrom (list)
  (equal list (reverse-list list))
)
```

```
> (is-palindrom `( 1 2 2 1) ) → T
> (is-palindrom `( 1 2 2 4) ) → Nil
```

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

```
(defun swap-first-last (list)
  (reduce
    #'(lambda (result tmp)
      (cond ((equal (length result) (- (length list) 2))
        (append (append (list tmp) result) (list (car list))))
        (t (append result (list tmp)))
      )
    ) (cdr list) :initial-value nil
  )
)
> ( swap-first-last `(1 2 3 4)) → (4 2 3 1)
```

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

```
(defun my-member (n list)
  (reduce
    (lambda (result tmp)
      (if (equal (length result) (- (length list) n))
        result
        (cdr result)
      )
    ) list :initial-value list
  )
)
```

```
(defun get-by-number (n list)
  (car (my-member n list))
)
```

```

(defun swap-two-ellement (left right list)
  (reduce
    (lambda (result tmp)
      (cond
        (
          (equal (length result) left)
          (append result (list (get-by-number right list)))
        )

        (
          (equal (length result) right)
          (append result (list (get-by-number left list)))
        )

        (
          t
          (append result (list tmp))
        )
      )
    ) list :initial-value nil
  )
)

```

> (swap-two-ellement 0 3 '(1 2 3 4)) → (4 2 3 1)

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

```

(defun swap-to-left (lst)
  (reduce
    (lambda (tmp result)
      (append (list tmp) result))
    (cdr lst) :initial-value (list (car lst)) :from-end t
  )
)

```

```

(defun swap-to-right (list)
  (reduce
    (lambda (result tmp)
      (if (equal (length result) (length list))
        result
        (append result (list tmp))
      )
    ) list :initial-value (last list)
  )
)

```

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

а) все элементы списка --- числа,

```
(defun mulall (mul lst)
  (mapcar (lambda (x) (* x mul)) lst)
)
```

> (mulall 4 (1 2 3 4)) → (4 8 12 16)

6) элементы списка -- любые объекты.

```
(defun mulall_2 (mul lst)
  (mapcar #'(lambda (x)
    (cond
      ((numberp x) (* x mul))
      ((listp x) (mulall_2 mul x))
      (t x)))
    lst
  )
)
```

> (mulall_2 2 (1 (2 (3)) 4)) → (2 (4 (6)) 8)

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_inner (lst left right result)
  (mapcar #'(lambda (x)
    (cond
      ((and (> x left) (< x right))
        (nconc result (cons x nil)))
      (t x)))
    lst
  )
  (cdr result)
)

(defun getfromto_2 (lst left right);
  (select_between_inner lst left right (cons nil nil))
)
```

```
(defun insert-elem (elem list)
  (cond
    ((null list) (cons elem nil))
    ((< elem (car list)) (cons elem list))
    (t (cons (car list) (insert-elem elem (cdr list)))))
)
```

)

```
(defun my-sort (list)
  (reduce
    (lambda (sorted tmp)
      (insert-elem tmp sorted))
    list :initial-value nil)
  )
)
```

```
(defun super-select-between(left right list)
  (my-sort (getfromto_2 list left right)))
```

> (print (super-select-between 4 8 '(1 2 3 4 5 6 7 5 5 5 6 6 2 2))) → (5 5 5 5 6 6 6 7)