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

Отчет к лабораторной работе №6

по курсу «Функциональное и логическое программирование»

по теме «Использование функционалов»

Студент: Костев Д.И.

Группа: ИУ7-61Б

Преподаватель: Толпинская Н.Б.

2022 г.

Практические задания

1. Напишите функцию, которая уменьшает на 10 все числа из списка-аргумента этой функции.

Решение:

(defun minus_10 (lst) (mapcar '(lambda (el) (setf el (- el 10))) lst))

Результат работы: (minus_10 '(11 12 13 14 15)) → (1 2 3 4 5)

2. 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

Решение:

а) (defun mult_numbers (lst num)

(mapcar '(lambda (el) (setf el (* el num))) lst))

Результат работы:

(mult_numbers '(1 2 3) 5) → (5 10 15)

б) (defun mult_various (lst num) (mapcar '(lambda (el) (if (numberp el)

(setf el (* el num)) el)) lst))

Результат работы:

(mult_various '(1 'abc '(3 4) 6 10 'abc '(1 b)) 2) → (2 'ABC '(3 4) 12 20 'ABC '(1 B))

3. Написать функцию, которая по своему списку-аргументу lst определяет является ли он палиндромом (то есть равны ли lst и (reverse lst)).

Решение:

(defun is_palyndrome (lst)

(every '(lambda (el) (not (eql el NIL)))

(mapcar '(lambda (el1 el2) (eql el1 el2)) lst (reverse lst))))

Результат работы:

(is_palyndrome '(1 2 3 2 1)) → T

(is_palyndrome '(1 2 3 2 4)) → NIL

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

Решение:

```
(defun set_equal (lst1 lst2) (if (= (length lst1) (length lst2))
        (every '(lambda (el) (not (eql el NIL)))
        (mapcar '(lambda (el) (if (member el lst1) T)) lst2))))
```

Результат работы:

(set_equal '(1 2 3) '(3 1 2)) → T

(set_equal '(1 2 3) '(3 4 2)) → NIL

(set_equal '(1 2 3) '(3 1 4 2)) → NIL

5. Написать функцию которая получает как аргумент список чисел, а возвращает список квадратов этих чисел в том же порядке.

Решение:

```
(defun return_square (lst) (mapcar '(lambda (el) (* el el)) lst))
```

Результат работы:

(return_square '(1 2 3)) → (1 4 9)

6. Напишите функцию, select-between, которая из списка-аргумента, содержащего толькочисла, выбирает только те, которые расположены между двумя указанными границамиаргументами и возвращает их в виде списка (упорядоченного по возрастанию списка чисел(+2 балла)).

Решение:

```
(defun select_between (lst a b)
    (funcall '(lambda (el) (remove NIL el))
    (mapcar '(lambda (el) (if (and (> el a) (> b el)) el)) lst)))
```

Результат работы:

(select_between '(1 2 3 4 5 6) 2 5) → (3 4)

(select_between '(1 2 3 4 5 6) 2 3) → NIL

7. Написать функцию, вычисляющую декартово произведение двух своих списков аргументов. (Напомним, что AxB это множество всевозможных пар (a b), где a принадлежит A, принадлежит B)

Решение:

```
(defun decart_prod (lst1 lst2)
    (mapcan '(lambda (x) (mapcar '(lambda (y) (list x y)) lst2)) lst1))
```

Результат работы:

(decart_prod '(1 2 3) '(4 5)) → ((1 4) (1 5) (2 4) (2 5) (3 4) (3 5))

8. Почему так реализовано reduce, в чем причина?

(reduce #'+0) -> 0

(reduce #'+ ()) -> 0

Решение:

Функция reduce сводит последовательность в одно значение. Она принимает функцию (она должна работать как минимум с двумя аргументами) и последовательность. Заданная функция первоначально применяется к первому элементу последовательности и initial-value, либо первым двум элементам, а затем последовательно к полученному результату и следующему элементу

последовательности. Последнее полученное значение будет возвращено как результат работы функции reduce.

Вызов (reduce 'func '(a b c d)) эквивалентен (func (func (func 'a 'b) 'c) 'd)

Initial-value для + и * - нейтральное (т.е. такое, что если к нему прибавить 0 или умножить на него 0, то ничего не изменится), следовательно: для + - это 0, для * - это 1.

Вызов (reduce '+ 0) -> 0 эквивалентен (+ 0 0) → 0

Функция + - это функционал, который при нулевом количестве аргументов возвращает inital-value, т.е. 0.

Следовательно (reduce '+ ()) -> 0

9. Пусть list-of-list список, состоящий из списков. Написать функцию, которая вычисляет сумму длин всех элементов list-of-list, т.е. например для аргумента ((1 2) (3 4)) -> 4.

Решение:

```
(defun length_sum (lst)
(reduce '+ (mapcar '(lambda (el) (length el)) lst)))
```

Результат работы:

(length_sum '((1 2) (3 4))) → 4

(length_sum '((1 2))) → 2

(length_sum '(())) → 0