|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 №9*

*По курсу: «Функциональное и логическое программирование»*

Студент ИУ7-66и

Ятагани Клейди

Преподаватель

Толпинская Н.Б

*Москва, 2020 г.*

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

; проверяет находится ли элемент в списке

(defun el-in-lst (el lst)

(cond

((null lst) nil)

((eq el (car lst)) t)

(t (el-in-lst el (cdr lst)))))

; проверяет находятся ли все элементы lst1 в lst2

(defun lst-in-lst (lst1 lst2)

(cond

((null lst1) t)

((el-in-lst (car lst1) lst2) (lst-in-lst (cdr lst1) lst2))

(t nil)))

(defun set-equal (lst1 lst2)

(cond

((and (null lst1) (null lst2)) t)

((or (null lst1) (null lst2)) nil)

(t (and (lst-in-lst lst1 lst2) (lst-in-lst lst2 lst1)))))

**2. Напишите необходимые функции, которые обрабатывают таблицу из точечных пар: (страна. столица), и возвращают по стране - столицу, а по столице - страну.**

(defun fb1 (capital lst)

(cond

((null lst) nil)

((eq capital (cdar lst)) (caar lst))

(T (fb1 capital (cdr lst))) ) )

;(print (fb1 'moscow (fb c s))) ; russia

(defun fb2 (country lst)

(cond

((null lst) nil)

((eq country (caar lst)) (cdar lst))

(T (fb2 country (cdr lst))) ) )

;(print (fb2 'russia (fb c s))) ; moscow

**3. Написать функцию, которая возвращает первый аргумент списка -аргумента. который сам является непустым списком.**

Исходя из задания:

(defun f3 (lst)

(car lst))

Следуя логике:

(defun f3 (lst)

(cond

((null lst) nil)

((atom lst) lst)

(t (f3 (car lst)))))

(print (f3 '(((1 2) 3) (4 5)))) ; 1

**4. Написать функцию, которая выбирает из заданного списка только те числа, которые больше 1 и меньше 10. (Вариант: между двумя заданными границами)**

(defun between\_x\_y (lst x y &optional res)

(cond

((null lst) res)

((< x (car lst) y) (between\_x\_y (cdr lst) x y (cons (car lst) res)))

(t (between\_x\_y (cdr lst) x y res))))

(print (between\_x\_y '(1 2 3 4 5 6 7 8 9 10 11) 3 7)) ; (6 5 4)

**5. Написать функцию, вычисляющую декартово произведение двух своих списков аргументов. (Напомним, что А х В — это множество всевозможных пар (a b), где а принадлежит А, b принадлежит В)**

; перемножаем отдельный элемент со списком

(defun mult\_el (el lst2 &optional res)

(cond

((null lst2) res)

(t (mult\_el el (cdr lst2) (cons (list el (car lst2)) res)))))

; перемножаем списки

(defun mult\_lsts (lst1 lst2 &optional res)

(cond

((null lst1) res)

(t (mult\_lsts (cdr lst1) lst2 (cons (mult\_el (car lst1) lst2) res)))))

(print (mult\_lsts '(a b) '(1 2))) ; (((b 2) (b 1) ((a 2) (a 1)))

**6. Почему так реализовано reduce, в чем причина?**

**(reduce #’\* ()) -> 1**

**(reduce #’+ ()) -> 0**

**(reduce #’append ()) -> nil**Начальный результат в reduce -–нейтральный: для суммы это 0 (если прибавить 0 ничего не изменится), для умножения – 1.

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

**a) все элементы списка --- числа,**

(defun mult-lst-inner (lst el &optional res)

(cond

((null lst) res)

(t (mult-lst-inner (cdr lst) el (cons (\* (car lst) el) res)))))

(defun mult-lst (lst el)

(reverse (mult-lst-inner lst el)))

(print (mult-lst '(1 2 3 4) 5)) ; (5 10 15 20)

**6) элементы списка -- любые объекты.**

(defun mult-lst-inner (lst el &optional res)

(cond

((null lst) res)

((numberp (car lst)) (mult-lst-inner (cdr lst) el (cons (\* (car lst) el) res)))

(t (mult-lst-inner (cdr lst) el (cons (car lst) res)))))

(defun mult-lst (lst el)

(reverse (mult-lst-inner lst el)))

(mult-lst ‘(1 2 3 a 4 b) 5) ; (5 10 15 a 20 b)