Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 6
По курсу «Функциональное и логическое програмирование».

Работа с функционалами

Студент Степанов А.О.
Группа ИУ7-63Б
Преподаватель Толпинская Н.Б.

Москва, 2020 г.

ЗАДАНИЕ 1

Дано два списка: первый список название стран, второй – столиц.

— из двух списков создать список из двухэлементных списков

— из двух списков создать список из точечных пар

По полученным спискам по стране найти столицу и наоборот.

```
(setq countries '(Russia USA GB Belarus))
(setq cities '(Moscow Washington London Minsk))
```

СПОСОБ 1: СПИСКИ

Создание списка, состоящего из двухэлементных списков.

```
(defun list_merge (countries cities)
    (mapcar #'(lambda (ctr cty) (list ctr cty)) countries cities)
)

(setq list_cc (list_merge countries cities))
```

Поиск страны по столице в списке из двухэлементных списков.

```
(defun list_country (list_cc city)
    (reduce #'(lambda (a b) (or a b))
        (mapcar #'(lambda (el)
                    (and (equal (cadr el) city) (car el))
                ) list_cc
        )
    )
)

(list_country list_cc 'London) ;;; GB
(list_country list_cc 'NotExist) ;;; Nil
(list_country list_cc 'Moscow) ;;; Russia
```

Поиск столицы по стране в списке из двухэлементных списков.

```
(defun list_city (list_cc country)
    (reduce #'(lambda (a b) (or a b))
        (mapcar #'(lambda (el)
                    (and (equal (car el) country) (cadr el))
                ) list_cc
```

```
        )
    )
)

(list_city list_cc 'GB) ;;; London
(list_city list_cc 'NotExist) ;;; Nil
(list_city list_cc 'Russia) ;;; Moscow
```

СПОСОБ 2: ТОЧЕЧНЫЕ ПАРЫ

Создание списка, состоящего из точечных пар.

```
(defun cons_merge (countries cities)
        (mapcar #'(lambda (ctr cty) (cons ctr cty)) countries cities)
)

(setq cons_cc (cons_merge countries cities))
```

Поиск страны по столице в списке из двухэлементных списков.

```
(defun cons_country (cons_cc city)
    (reduce #'(lambda (a b) (or a b))
        (mapcar #'(lambda (el)
                    (and (equal (cdr el) city) (car el))
                ) cons_cc
        )
    )
)

(cons_country cons_cc 'London) ;;; GB
(cons_country cons_cc 'NotExist) ;;; Nil
(cons_country cons_cc 'Moscow) ;;; Russia
```

Поиск столицы по стране в списке из двухэлементных списков.

```
(defun cons_city (cons_cc country)
    (reduce #'(lambda (a b) (or a b))
        (mapcar #'(lambda (el)
                    (and (equal (car el) country) (cdr el))
                ) cons_cc
        )
    )
)

(cons_city cons_cc 'GB) ;;; London
```

```
(cons_city cons_cc 'NotExist) ;;; Nil
(cons_city cons_cc 'Russia) ;;; Moscow
```

ВЫВОДЫ

Удобней и эффективней использовать второй способ со списком, состоящим из точечных пар, так как для обращения к второму элементу пары необходимо использовать на одну функцию меньше. Также при использовании списка, состоящего из двухэлементных списков на каждую пару выделяется три списковых ячейки, а во втором случае – только две.

ЗАДАНИЕ 2

```
(defun new_how_alike (x y)
    (if (or (= x y) (equal x y))
        'the_same
        (if (and (oddp x) (oddp y))
            'both_odd
            (if (and (evenp x) (evenp y))
                'both_even
                'difference))))

(new_how_alike 2 4) ;;; both_even
(new_how_alike 3 3) ;;; the_same
(new_how_alike 3 5) ;;; both_odd
(new_how_alike 4 5) ;;; difference
```