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».

Рекурсивные функции

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_cc (countries cities)
    (cond
        ((null countries)  Nil)
        ((null cities)     Nil)
        (T (cons
            (list (car countries) (car cities))
            (list_merge_cc (cdr countries) (cdr cities))
        ))
    )
)

(setq list_merged_cc (list_merge_cc countries cities))
```

Поиск страны по столице в списке из двухэлементных списков.

```
(defun list_find_country (list_cc city)
    (cond
        ((null list_cc) Nil)
        ((equal (cadar list_cc) city) (caar list_cc))
        (T (list_find_country (cdr list_cc) city))
    )
)

(list_find_country list_merged_cc 'Moscow) ;;; Russia
(list_find_country list_merged_cc  'Minsk) ;;; Belarus
(list_find_country list_merged_cc 'Sidney) ;;; Nil
```

Поиск столицы по стране в списке из двухэлементных списков.

```
(defun list_find_city (list_cc country)
    (cond
        ((null list_cc) Nil)
        ((equal (caar list_cc) country) (cadar list_cc))
        (T (list_find_city (cdr list_cc) country))
    )
)

(list_find_city list_merged_cc    'Russia) ;;; Moscow
(list_find_city list_merged_cc   'Belarus) ;;; Minsk
(list_find_city list_merged_cc 'Australia) ;;; Nil
```

СПОСОБ 2: ТОЧЕЧНЫЕ ПАРЫ

Создание списка, состоящего из точечных пар.

```
(defun cons_merge_cc (countries cities)
    (cond
        ((null countries)  Nil)
        ((null cities)     Nil)
        (T (cons
            (cons (car countries) (car cities))
            (cons_merge_cc (cdr countries) (cdr cities))
        ))
    )
)

(setq cons_merged_cc (cons_merge_cc countries cities))
```

Поиск страны по столице в списке из двухэлементных списков.

```
(defun cons_find_country (list_cc city)
    (cond
        ((null list_cc) Nil)
        ((equal (cdar list_cc) city) (caar list_cc))
        (T (cons_find_country (cdr list_cc) city))
    )
)

(cons_find_country cons_merged_cc 'Moscow) ;;; Russia
(cons_find_country cons_merged_cc  'Minsk) ;;; Belarus
(cons_find_country cons_merged_cc 'Sidney) ;;; Nil
```

Поиск столицы по стране в списке из двухэлементных списков.

```
(defun cons_find_city (list_cc country)
    (cond
        ((null list_cc) Nil)
        ((equal (caar list_cc) country) (cdar list_cc))
        (T (cons_find_city (cdr list_cc) country))
    )
)

(cons_find_city cons_merged_cc    'Russia) ;;; Moscow
(cons_find_city cons_merged_cc   'Belarus) ;;; Minsk
(cons_find_city cons_merged_cc 'Australia) ;;; Nil
```

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
```