ВВЕДЕНИЕ

Целью работы является приобретение навыков использования списков и стандартных функций Lisp.

Задачи работы: изучить способ использования списков для фиксации информации, внутреннее представление одноуровневых и структурированных списков, методы их обработки с использованием базовых функций Lisp.

1 Практическая часть

1.1 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))
```

1.1.1 Списки

Создание списка из двухэлементных списков:

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

(setq list_cc (create_list 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
        )
    )
)
```

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
        )
    )
)
```

1.1.2 Точеные пары

Создание списка, состоящего из точечных пар:

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

(setq cons_cc (create_cons 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
        )
    )
)
```

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
        )
    )
)
```

1.2 Задание №2

Переписать функцию how-alike, приведенную в лекции и не пользующую COND, используя конструкция IF, AND/OR.

```
(defun how-alike (x y)
    (if (or (= x y) (equal x y))
            'sameNumbers
            (if (and (oddp x) (oddp y))
                'oddpNumbers
                (if (and (evenp x) (evenp y))
                    'evenpNumbers
                    'differenceNumbers
                )
```

```
10                          )
11              )
12    )
```