个学生排成一排，学生编号分别是 1 到 n，n 为 3 的整倍数。老师随机抽签决定将所有学生分成 m 个 3 人的小组，n=3*m
为了便于同组学生交流，老师决定将小组成员安排到一起，也就是同组成员彼此相连，同组任意两个成员之间无其它组的成员。
因此老师决定调整队伍，老师每次可以调整任何一名学生到队伍的任意位置，计为调整了一次，
请计算最少调整多少次可以达到目标。
注意：对于小组之间没有顺序要求，同组学生之间没有顺序要求。
输入描述：
两行字符串，空格分隔表示不同的学生编号
第一行是学生目前排队情况
第二行是随机抽签分组情况，从左开始每 3 个元素为一组
n 为学生的数量， n 的范围为[3, 900]，n 一定为 3 的整倍数
第一行和第二行的元素个数一定相同

输出描述：
老师调整学生达到同组彼此相连的最小次数

补充说明：
同组相连： 同组任意两个成员之间无其它组的成员 ，比如有两个小组[4 5 6] [1 2 3],以下结果都满足要求
1 2 3 4 5 6
1 3 2 4 5 6
2 3 1 5 6 4
5 6 4 1 2 3
以下结果不满足要求
1 2 4 3 5 6， 4 与 5 之间存在其它组的成员 3

示例 1
输入：
7 9 8 5 6 4 2 1 3
7 8 9 4 2 1 3 5 6
输出：
1
说明：
学生目前排队情况: 7 9 8 5 6 4 2 1 3
学生分组情况：[7 8 9] [4 2 1] [3 5 6]
将 3 调整到 4 之前，队列调整为 7 9 8 5 6 3 4 2 1，那么三个小组成员均彼此相连[7 9 8] [5 6 3] [4 2 1]
输出：1
示例 2
输入：
8 9 7 5 6 3 2 1 4

7 8 9 4 2 1 3 5 6
输出：
0
说明：
学生目前排队情况：7 9 8 5 6 3 2 1 4
学生分组情况：[7 8 9] [4 2 1] [3 5 6]
无需调整，三个小组成员均彼此相连[7 9 8] [5 6 3] [2 1 4]
输出：0
代码：

```
using System.Collections.Generic;
public class Program {
    public static void Main() {
        string line;
        while ((line = System.Console.ReadLine ()) !=
                null) { // 注意 while 处理多个 case
            string[] tokens = line.Split((char)32);
            int groupNum = tokens.Length / 3;
            List<int> originList = new List<int>();
            for (int i = 0; i < tokens.Length; i++) {
                originList.Add(int.Parse(tokens[i]));
            }
            line = System.Console.ReadLine();
            tokens = line.Split((char)32);
            List<int[]> groupArr = new List<int[]>();
            for (int i = 0; i < groupNum; i++) {
                groupArr.Add(new int[3] {
                    int.Parse( tokens[i * 3]),
                    int.Parse( tokens[i * 3 + 1]),
                    int.Parse( tokens[i * 3 + 2])
                });
            }

            int result = 0;

            for (int i = 0; i < groupNum; i++) {
                int first = originList[i * 3];
                int second = originList[i * 3 + 1];
                int third = originList[i * 3 + 2];

                int row = 0;
                for (; row < groupNum; row++) {
                    if (groupArr[row][0] == first) {
                        break;
                    } else if (groupArr[row][1] == first) {
```

```
                    break;
                } else if (groupArr[row][2] == first) {
                    break;
                }
            }

            if (exist_2(groupArr[row], second, third)) {
                continue;
            } else if (exist_1(groupArr[row], second)) {
                result++;
                int tmpNum = find_1(groupArr[row], first, second);
                originList.Remove(tmpNum);
                originList.Insert(i * 3 + 2, tmpNum);
            } else {
                result += 2;
                int[] tmpArr = find_2(groupArr[row], first);
                for (int j = 0; j < 2; j++) {
                    int tmpNum = tmpArr[j];
                    originList.Remove(tmpNum);
                    originList.Insert(i * 3 + j + 1, tmpNum);
                }
            }

        }

        System.Console.WriteLine(result);
    }
}

static bool exist_1(int[] arr, int num) {
    for (int i = 0; i < arr.Length; i++) {
        if (arr[i] == num) {
            return true;
        }
    }
    return false;
}

static bool exist_2(int[] arr, int num1, int num2) {
    bool exist1 = false;
    bool exist2 = false;
    for (int i = 0; i < arr.Length; i++) {
        if (!exist1 && arr[i] == num1) {
            exist1 = true;
```

```
                }
                if (!exist2 && arr[i] == num2) {
                        exist2 = true;
                }
                if (exist1 && exist2) {
                        break;
                }
        }
        return exist1 && exist2;
    }

    static int find_1(int[] arr, int num1, int num2) {
        for (int i = 0; i < arr.Length; i++) {
                if (arr[i] != num1 && arr[i] != num2) {
                        return arr[i];
                }
        }
        return -1;
    }

    static int[] find_2(int[] arr, int num) {
        int[] result = new int[2];
        int j = 0;
        for (int i = 0; i < arr.Length; i++) {
                if (arr[i] != num) {
                        result[j++] = arr[i];
                }
        }
        return result;
    }

}
```