Assignment 1 – Letter order

```csharp
using System;

namespace Assignment1_LetterOrder
{
    class Program
    {
        static void Main(string[] args)
        {
            Program myProgram = new Program();
            myProgram.Start();
        }

        void Start()
        {
            Console.Write("Enter a sentence: ");
            string sentence = Console.ReadLine();

            // shuffle words
            string newSentence = ShuffleWords(sentence);
            Console.WriteLine("The new sentence has become: {0}", newSentence);

            // wait for user
            Console.ReadKey();
        }

        string ShuffleWords(string sentence)
        {
            string shuffledSentence = "";
            string[] words = sentence.Split(' ');

            foreach (string word in words)
            {
                string shuffledWord = ShuffleWord(word);
                shuffledSentence += shuffledWord + " ";
            }

            return shuffledSentence;
        }
        string ShuffleWord(string word)
        {
            // "according" => "a.......g"
            // "research" => "r......h"

            if (word.Length <= 3)
                return word;

            Random rnd = new Random();

            string newWord = "";

            // copy first letter
            newWord += word[0];

            // shuffle middle part
            string remainingWord = word.Substring(1, word.Length - 2);
            while (remainingWord != "")
            {
                // copy a random letter from remaining word
                int index = rnd.Next(remainingWord.Length);
                newWord = newWord + remainingWord[index];

                // remove letter from remaining word
                remainingWord = remainingWord.Remove(index, 1);
            }

            // copy last letter
            newWord = newWord + word[word.Length - 1];

            return newWord;
        }
    }
}
```

Assignment 2 – Shift matrix

```
using System;

namespace Assignment2_ShiftMatrix
{
    class Program
    {
        static void Main(string[] args)
        {
            Program myProgram = new Program();
            myProgram.Start();
        }

        void Start()
        {
            int[,] matrix = new int[5, 10];    // 5 rows, 10 columns

            FillMatrix(matrix);
            DisplayMatrix(matrix);

            Console.WriteLine();
            Console.Write("Enter a new number: ");
            int number = Int32.Parse(Console.ReadLine());
            Console.WriteLine();

            ShiftMatrix(matrix, number);
            Console.WriteLine();
            DisplayMatrix(matrix);

            // wait for user
            Console.ReadKey();
        }

        void FillMatrix(int[,] matrix)
        {
            Random rnd = new Random();
            for (int row = 0; row < matrix.GetLength(0); row++)
            {
                for (int col = 0; col < matrix.GetLength(1); col++)
                {
                    matrix[row, col] = rnd.Next(1, 100);
                }
            }
        }

        void DisplayMatrix(int[,] matrix)
        {
            for (int row = 0; row < matrix.GetLength(0); row++)
            {
                for (int col = 0; col < matrix.GetLength(1); col++)
                {
                    Console.Write("{0:00} ", matrix[row, col]);
                }
                Console.WriteLine();
            }
        }
        void ShiftMatrix(int[,] matrix, int number)
        {
            for (int row = 0; row < matrix.GetLength(0); row++)
            {
                // search number in row
                for (int col = 0; col < matrix.GetLength(1); col++)
                {
                    if (matrix[row, col] == number)
                    {
                        Console.WriteLine("shift row {0} starting from column {1}...",
                                          row + 1, col + 1);
                        ShiftRow(matrix, row, col);
                        break;
                    }
                }
            }
        }

        void ShiftRow(int[,] matrix, int row, int column)
        {
            int[] temp = new int[matrix.GetLength(1)];

            // store shifted numbers in temp array
            for (int col = 0; col < matrix.GetLength(1); col++)
            {
                temp[col] = matrix[row, column];
                column = (column + 1) % matrix.GetLength(1);
            }

            // store numbers in temp-array back to matrix
            for (int col = 0; col < matrix.GetLength(1); col++)
            {
                matrix[row, col] = temp[col];
            }
        }
    }
}
```

Assignment 3 – Election

[file Premise.cs]
```
namespace Assignment3_Election
{
    public class Premise
    {
        public string title;
        public string text;
    }
}
```

[file Party.cs]
```
namespace Assignment3_Election
{
    public class Party
    {
        public string name;
        public string answers;  // 30 characters (1|2|3)
    }
}
```

[file Program.cs]
```
using System;
using System.Collections.Generic;
using System.IO;

namespace Assignment3_Election
{
    class Program
    {
        const string PREMISES_FILENAME = "premises.txt";
        const string PARTIES_FILENAME = "parties.txt";

        static void Main(string[] args)
        {
            Program myProgram = new Program();
            myProgram.Start();
        }

        void Start()
        {
            // read premises
            List<Premise> premises = ReadPremises(PREMISES_FILENAME);
            if (premises.Count == 0)
            {
                Console.WriteLine("No premises read...");
                return;
            }

            // read parties
            List<Party> parties = ReadParties(PARTIES_FILENAME);
            if (parties.Count == 0)
            {
                Console.WriteLine("No parties read...");
                return;
            }
        // process all premises
        string userAnswers = ProcessPremises(premises);

        // compare user answers against answers of all parties
        CompareParties(userAnswers, parties);

        Console.ReadKey();
    }

    List<Premise> ReadPremises(string filename)
    {
        List<Premise> premises = new List<Premise>();
        if (!File.Exists(filename))
            return premises;

        StreamReader reader = new StreamReader(filename);
        while (!reader.EndOfStream)
        {
            Premise premise = new Premise();
            premise.title = reader.ReadLine();
            premise.text = reader.ReadLine();
            premises.Add(premise);
        }
        reader.Close();

        return premises;
    }

    List<Party> ReadParties(string filename)
    {
        List<Party> parties = new List<Party>();
        if (!File.Exists(filename))
            return parties;

        StreamReader reader = new StreamReader(filename);
        while (!reader.EndOfStream)
        {
            Party party = new Party();
            party.name = reader.ReadLine();
            party.answers = reader.ReadLine();
            parties.Add(party);
        }
        reader.Close();

        return parties;
    }
```

```
        string ProcessPremises(List<Premise> premises)
        {
            string anwers = "";

            // process all premises
            int nr = 1;
            foreach (Premise premise in premises)
            {
                // display premise
                Console.ForegroundColor = ConsoleColor.Yellow;
                Console.WriteLine("{0}. {1}", nr, premise.title);
                Console.ResetColor();
                Console.WriteLine(premise.text);
                Console.WriteLine();

                // ask opinion of user
                Console.Write("Enter your opinion (1=agree / 2=disagree / 3=no
opinion): ");
                string opinion = Console.ReadLine();
                Console.WriteLine();

                anwers = anwers + opinion;

                nr++;
            }
            return anwers;
        }

        void CompareParties(string user, List<Party> parties)
        {
            // compare all parties and user
            foreach (Party party in parties)
            {
                double partyPerc = DetermineMatch(user, party);

                // display percentage
                Console.WriteLine("{0,-5}: {1:0.0} %", party.name, partyPerc);
            }
        }

        double DetermineMatch(string user, Party party)
        {
            // compare all answers (user and party)
            int points = 0;
            for (int i = 0; i < user.Length; i++)
            {
                if (party.answers[i] == user[i])
                {
                    points++;
                }
            }
            return (points / (double)user.Length) * 100;
        }
    }
}
```