```csharp
/// PROGRAMMER : Jonas Smith
/// Purpose    : Get inputs from a user and solve a given linear equation.
/// Resources  : https://numerics.mathdotnet.com/LinearEquations.html - resource 
  to make this more general case with Math.net
///            :
///


using System;
using Project_1.classes;
using System.Collections.Generic;
using MathNet.Numerics.LinearAlgebra;

namespace Project_01
{
    class ProjectOne
    {
        static void Main()
        {
            bool user_is_iterating = true;
            List<char> variable_names = new List<char>() { 'a', 'b', 'c', 'd', 
              'r', 's' };
            List<char> solution_names = new List<char>() { 'x', 'y' };
            int row_n = 2;
            int col_n = 2;

            TestCases test_cases = new TestCases();

            // flip this bit to begin testing
            bool testing;
            int test_index = 0;

            testing = PromptForTesting();
            Console.Clear();

            PrintHeader();
            PrintPurpose();
            PrintDivider();

            if (testing)
            {
                Console.WriteLine("Press any key while testing");
            }

            while (user_is_iterating)
            {
                List<variable> user_inputs = new List<variable>();

                if (testing)
                {
                    if (test_index >= test_cases.cases.Count)
                    {
                        goto stopTesting;
                    }

                    user_inputs = test_cases.cases[test_index];
                    test_index++;
                }
                else
                {
                    user_inputs = GetUserInput(variable_names);
                }

                Vector<double> solution = CalculateSolutions(user_inputs, row_n, col_n);

                PrintSolutions(solution, solution_names, user_inputs, testing);
                UserIterations(ref user_is_iterating, testing);
            }

            stopTesting:;
            if (testing)
            {
                Console.WriteLine("      Testing is finished");
                Console.ReadLine();
            }
        }

        static bool PromptForTesting()
        {
            bool correct_input = false;

            while (!correct_input)
            {
                Console.Clear();
                Console.WriteLine("Would you like to use the test data?");
                Console.Write("           [y]es or [n]o : ");

                string user_input = Console.ReadLine().ToLower();

                if (user_input == "y")
                {
                    return true;
                }
                else if (user_input == "n")
                {
                    return false;
                }
            }

            return false;
        }


```

```csharp
        static void PrintHeader()
        {
            List<string> headers = new List<string>() { "Math 371", "Spring 2020", "Lienar System Solver", "Jonas Smith" };
            int size = 32;
            string buffer = "";

            for (int i = 0; i < headers.Count; i++)
            {

                buffer = GetBuffer(headers[i], size);
                Console.WriteLine("{0}{1}", buffer, headers[i]);
            }

        }

        static string GetBuffer(string value, int length)
        {
            int buffer_length = (length - Convert.ToInt32(value.Length)) / 2;
            string buffer = "";

            for (int i = 0; i < buffer_length; i++)
                buffer += " ";

            return buffer;
        }

        static void PrintPurpose()
        {
            Console.WriteLine();
            Console.WriteLine("     Take inputs from the user ");
            Console.WriteLine("{a,b,c,d,r,s} and calculate x");
            Console.WriteLine("and y from the system of linear");
            Console.WriteLine("equations");
            Console.WriteLine();
            Console.WriteLine("similar to: ax + by = r");
            Console.WriteLine("            cx + dy = s");

        }

        static void PrintDivider() => Console.WriteLine("________________________________\n");

        /// <summary>
        ///     Prompts the user to enter a variable based on the list given
        /// </summary>
        static List<variable> GetUserInput(List<char> variables)
        {
            List<variable> user_inputs = new List<variable>();

            for (int i = 0; i < variables.Count; i++)
            {
                user_inputs.Add(GetVariableInput(variables[i], i));
            }

            return user_inputs;
        }

        /// <summary>
        /// Only allows numbers . and - to be entered into the fields
        /// </summary>
        /// <param name="var_name"></param>
        /// <param name="index"></param>
        /// <returns></returns>
        static variable GetVariableInput(char var_name, int index)
        {
            bool user_input_wrong = true;

            string message = "   Enter the value for";

            string buffer = "";

            for (int i = 0; i < message.Length; i++)
            {
                buffer += " ";
            }

            double input = 0.0;

            while (user_input_wrong)
            {
                string prompt = string.Format("{0} {1} = ", buffer, var_name);

                if (index == 0)
                {
                    prompt = string.Format("{0} {1} = ", message, var_name);
                }

                Console.Write(prompt);

                try
                {
                    string _val = "";

                    ConsoleKeyInfo key;

                    do
                    {
                        key = Console.ReadKey(true);
                        if (key.Key != ConsoleKey.Backspace)
                        {
                            double val = 0;
                            bool _x = double.TryParse(key.KeyChar.ToString(), out ↩
                              val);
                            if (_x)
                            {
                                _val += key.KeyChar;
                                Console.Write(key.KeyChar);
                            }

                            if (key.Key == ConsoleKey.OemPeriod)
                            {
                                _val += ".";
                                Console.Write(key.KeyChar);
                            }

                            if (key.Key == ConsoleKey.OemMinus)
                            {
                                _val += "-";
                                Console.Write(key.KeyChar);
                            }
                        }
                        else
                        {
                            if (key.Key == ConsoleKey.Backspace && _val.Length > 0)
                            {
                                _val = _val.Substring(0, (_val.Length - 1));
                                Console.Write("\b \b");
                            }
                        }
                    } while (key.Key != ConsoleKey.Enter);


                    input = Convert.ToDouble(_val);

                    user_input_wrong = false;
                }
                catch
                {
                    index++;
                }

                Console.WriteLine();
            }


            return new variable(var_name, input);
        }

        /// <summary>
        ///     Moves the user_input variable list into a set of arrays used to calculate the values.
        /// </summary>
        /// <param name="user_input"></param>
        /// <returns></returns>
```

```csharp
        static Vector<double> CalculateSolutions(List<variable> user_input, int row_n, int col_n)
        {
            // Initialize the row and col number given when the application first runs.
            //      This can be changed easily enough with the lists to allow this to work
            //      with any number of linear equations and variables.
            int row = row_n;
            int col = col_n;

            // Using some linear algebra we can use Ax = b
            // Build matrix A
            double[,] matrix_A = new double[row, row];

            int index = 0;

            for (int i = 0; i < row; i++)
            {
                for (int k = 0; k < col; k++)
                {
                    matrix_A[i, k] = user_input[index].value;
                    index++;
                }
            }

            var A = Matrix<double>.Build.DenseOfArray(matrix_A);

            // Build the coefficient vector b
            double[] matrix_B = new double[row];

            for (int i = 0; i < col; i++)
            {
                matrix_B[i] = user_input[index].value;

                index++;
            }

            var b = Vector<double>.Build.Dense(matrix_B);

            // Solve!
            Vector<double> x = A.Solve(b);

            return x;
        }

        static void PrintSolutions(Vector<double> solutions, List<char> solution_names, List<variable> user_input, bool testing)
        {
            if (testing)
            {
                Console.WriteLine("\n         Solutions for         ");

                string varialbe_string = "";

                for (int i = 0; i < user_input.Count; i++)
                {

                    if (i != user_input.Count - 1)
                    {

                        varialbe_string += string.Format("{0}={1}, ", user_input[i].name, user_input[i].value);
                    }
                    else
                    {

                        varialbe_string += string.Format("and {0}={1}", user_input[i].name, user_input[i].value);
                    }
                }

                Console.WriteLine("{0}", varialbe_string);
            }
            else
            {
                Console.WriteLine("\n           Solutions           ");
                Console.WriteLine("________________________________");
            }

            for (int i = 0; i < solutions.Count; i++)
            {
                string output = "";

                // there is an infinite number of intersections.
                if (Double.IsNaN(solutions[i]))
                {
                    output = string.Format("{0} ={1}", solution_names[i], " Infinitely many");
                }
                // If the object is not a number therefore we know we have a zero in the denominator
                else if (Double.IsInfinity(solutions[i]))
                {
                    output = string.Format("{0} ={1}", solution_names[i], " No solution");
                }
                // one solution.
                else
                {
                    // the solutions is negative so we move the margin over one characters
                    if (solutions[i] < 0)
                        output = string.Format("{0} ={1}", solution_names[i],
                            solutions[i].ToString("N5"));
                    else
                        output = string.Format("{0} = {1}", solution_names[i], solutions[i].ToString("N5"));
                }

                string buffer = GetBuffer(output, 32);

                Console.WriteLine("{0}{1}", buffer, output);
            }

            if (testing)
                Console.WriteLine("________________________________");

            Console.WriteLine();
        }

        static void UserIterations(ref bool user_iteration, bool testing)
        {
            if (!testing)
            {
                bool correct_input = true;

                while (correct_input)
                {

                    Console.WriteLine("   Would you like to continue?");
                    Console.Write("        [y]es or [n]o   : ");

                    string user_input = Console.ReadLine().ToLower();

                    if (user_input == "y")
                    {
                        correct_input = false;
                        Console.Clear();
                        Iterate();
                    }
                    else if (user_input == "n")
                    {
                        user_iteration = false;
                        correct_input = false;
                    }
                    else
                    {
                        Console.Write(new String(' ', Console.BufferWidth));
                    }
                }
            }
            else
            {
                Console.ReadLine();
            }
        }

        static void Iterate()
        {
            PrintHeader();
            PrintPurpose();
            PrintDivider();
        }
    }
}

```