```
// Mekhi Prospere
// Program 4: CSI Find the Sample Game
// September 29th,2024
// This Windows Application is a  It involves a 2D grid where the
// player must guess the locations of two hidden evidence samples.

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;


namespace CSIFindEvidenceGame
{
    public partial class EvidenceGameForm : Form
    {
        private EvidenceScanner evidenceScanner;

        public EvidenceGameForm()
        {
            InitializeComponent();
            ShowInstructions();
        }

        private void ShowInstructions()
        {
            MessageBox.Show("Welcome to the Evidence Finder Game!\n" +
                            "Enter the dimensions for the search grid (max
                    10x10).\n" +
                            "Make your guesses to find the samples. Good luck!");
        }

        private void btnStartGame_Click(object sender, EventArgs e)
        {
            // Validate and parse grid size inputs
            int rows, columns;
            if (!int.TryParse(txtRows.Text, out rows) || !int.TryParse
              (txtColumns.Text, out columns) ||
                rows <= 0 || columns <= 0 || rows > 10 || columns > 10)
            {
                MessageBox.Show("Please enter valid dimensions (1-10). Defaulting
                   to 5x5.");
                rows = columns = 5;
            }

            evidenceScanner = new EvidenceScanner(rows, columns);
            txtGridDisplay.Text = evidenceScanner.DisplayGrid();
```

```
50              lblFeedback.Text = "Game started! Enter your first guess.";
51              lblGuesses.Text = "Number of guesses: 0";
52          }
53
54          private void btnSubmitGuess_Click(object sender, EventArgs e)
55          {
56              if (evidenceScanner == null || evidenceScanner.AllSamplesFound())
57              {
58                  MessageBox.Show("Game is over. Please restart to play again.");
59                  return;
60              }
61
62              int guessRow, guessCol;
63              if (!int.TryParse(txtGuessRow.Text, out guessRow) || !int.TryParse
                  (txtGuessCol.Text, out guessCol) ||
64                  guessRow < 0 || guessCol < 0 || guessRow >= evidenceScanner.Rows
                      || guessCol >= evidenceScanner.Columns)
65              {
66                  MessageBox.Show("Guess out of bounds! Please try again.");
67                  return;
68              }
69
70              bool sampleFound = evidenceScanner.EvaluateGuess(guessRow, guessCol);
71              txtGridDisplay.Text = evidenceScanner.DisplayGrid();
72
73              lblFeedback.Text = sampleFound ? "Sample found!" : "Keep looking!";
74              lblGuesses.Text = $"Number of guesses: {evidenceScanner.GuessCount}";
75
76              if (evidenceScanner.AllSamplesFound())
77              {
78                  lblFeedback.Text = "Success! You've found both samples.";
79                  btnRestart.Enabled = true;
80                  btnSubmitGuess.Enabled = false;
81              }
82          }
83
84          private void btnRestart_Click(object sender, EventArgs e)
85          {
86              txtGridDisplay.Clear();
87              lblFeedback.Text = "Game reset! Enter grid size and start again.";
88              btnSubmitGuess.Enabled = true;
89              btnRestart.Enabled = false;
90              txtRows.Clear();
91              txtColumns.Clear();
92              lblGuesses.Text = "Number of guesses: 0";
93              evidenceScanner = null; // Reset the game state
94          }
95
96          private void btnQuit_Click(object sender, EventArgs e)
97          {
98              if (evidenceScanner != null)
99              {
```

```
100                 MessageBox.Show($"Sample locations were:\n
                      {evidenceScanner.GetSampleLocations()}");
101             }
102             Application.Exit();
103         }
104     }
105
106     public class EvidenceScanner
107     {
108         private string[,] grid;
109         private int rows;
110         private int columns;
111         private int guessCounter;
112         private int[] sample1;
113         private int[] sample2;
114         private Random random;
115
116         public int Rows => rows;
117         public int Columns => columns;
118         public int GuessCount => guessCounter;
119
120         public EvidenceScanner(int rows, int columns)
121         {
122             this.rows = rows;
123             this.columns = columns;
124             this.grid = new string[rows, columns];
125             this.random = new Random();
126             InitializeGrid();
127             PlaceSamples();
128             guessCounter = 0;
129         }
130
131         private void InitializeGrid()
132         {
133             for (int i = 0; i < rows; i++)
134             {
135                 for (int j = 0; j < columns; j++)
136                 {
137                     grid[i, j] = "~"; // Initialize the grid with squiggly lines
138                 }
139             }
140         }
141
142         private void PlaceSamples()
143         {
144             sample1 = new int[] { random.Next(rows), random.Next(columns) };
145             do
146             {
147                 sample2 = new int[] { random.Next(rows), random.Next(columns) };
148             } while (sample1[0] == sample2[0] && sample1[1] == sample2[1]); //
                  Ensure different locations
149         }
```

```
150
151         public string DisplayGrid()
152         {
153             string result = "  ";
154             for (int i = 0; i < columns; i++)
155             {
156                 result += i.ToString();  // Column headers
157             }
158             result += "\n";
159
160             for (int i = 0; i < rows; i++)
161             {
162                 result += i.ToString() + " ";  // Row headers
163                 for (int j = 0; j < columns; j++)
164                 {
165                     result += grid[i, j];  // Grid content
166                 }
167                 result += "\n";
168             }
169
170             return result;
171         }
172
173         public bool EvaluateGuess(int guessRow, int guessCol)
174         {
175             guessCounter++;
176
177             if (guessRow == sample1[0] && guessCol == sample1[1] ||
178                 guessRow == sample2[0] && guessCol == sample2[1])
179             {
180                 grid[guessRow, guessCol] = "X";  // Mark sample found
181                 return true;
182             }
183             else
184             {
185                 // Provide directional feedback
186                 if (guessCounter % 2 == 0)  // Even guess: Check above/below
187                 {
188                     grid[guessRow, guessCol] = (guessRow > sample1[0] || guessRow
                        > sample2[0]) ? "^" : "v";
189                 }
190                 else  // Odd guess: Check left/right
191                 {
192                     grid[guessRow, guessCol] = (guessCol < sample1[1] || guessCol
                        < sample2[1]) ? ">" : "<";
193                 }
194
195                 return false;
196             }
197         }
198
199         public bool AllSamplesFound()
```

```
200         {
201             return grid[sample1[0], sample1[1]] == "X" && grid[sample2[0],
                  sample2[1]] == "X";
202         }
203
204         public string GetSampleLocations()
205         {
206             return $"Sample 1: ({sample1[0]}, {sample1[1]})\nSample 2: ({sample2
                  [0]}, {sample2[1]})";
207         }
208     }
209 }
210
```