Lab 08:

Code:

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 CCLab08

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void btnValidate\_Click(object sender, EventArgs e)

{

string input = txtVariable.Text;

lstTransitions.Items.Clear();

if (IsValidVariable(input))

{

lblResult.Text = "Valid variable!";

}

else

{

lblResult.Text = "Invalid variable!";

}

}

// DFA Logic to validate C variable names

private bool IsValidVariable(string input)

{

lstTransitions.Items.Add("Starting DFA Validation...");

if (string.IsNullOrEmpty(input))

{

lstTransitions.Items.Add("Input is empty.");

return false;

}

// Start state q0

if (!IsLetterOrUnderscore(input[0]))

{

lstTransitions.Items.Add("Invalid start character (must be a letter or underscore).");

return false;

}

lstTransitions.Items.Add("Start state: First character is valid.");

// Accepting state q1: Check subsequent characters

for (int i = 1; i < input.Length; i++)

{

if (!IsLetterOrDigitOrUnderscore(input[i]))

{

lstTransitions.Items.Add("Invalid character at position " + i + ": " + input[i]);

return false;

}

}

lstTransitions.Items.Add("All characters are valid.");

return true;

}

// Helper function to check if the character is a letter or underscore

private bool IsLetterOrUnderscore(char c)

{

return char.IsLetter(c) || c == '\_';

}

// Helper function to check if the character is a letter, digit or underscore

private bool IsLetterOrDigitOrUnderscore(char c)

{

return char.IsLetterOrDigit(c) || c == '\_';

}

}

}

Output:





