**Name: Anusha Amin**

**Reg No: FA20-BCS-032**

**Mid Lab (Compiler Construction)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question No.1**

A regular expression (regex) is an expression that contains one or many characters that expresses a given pattern in text.

**Example:**

If we have today’s date: 25-Oct-2023

Then using regex, we can express this format as: [0-9]{2}-[A-Z]{3}-[0-9]{4}

**Regex in C#:**

In C#, Regular Expressions are generally termed as C# Regex. The .Net Framework provides a regular expression engine that allows the pattern matching. Patterns may consist of any character literals, operators, or constructors. C# provides a class termed as Regex which can be found in System.Text.RegularExpression namespace.

This class will perform two things:

1. Parsing the inputting text for the regular expression pattern.
2. Identify the regular expression pattern in the given text.

**Example:**

**A simple example to check whether the email is valid or not.**

class Program {

static void Main() {

Pattern = @"^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$";

string validEmail = "example@email.com";

string invalidEmail = "invalid\_email.com";

Regex regex = new Regex(emailPattern);

Console.WriteLine($"Is '{validEmail}' a valid email? {regex.IsMatch(validEmail)}");

Console.WriteLine($"Is '{invalidEmail}' a valid email? {regex.IsMatch(invalidEmail)}");

}

}

**Question No.2**

**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 \_032\_MidLabQ2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

string input = textBox.Text;

LL1Parser parser = new LL1Parser(input);

parser.Parse();

if (parser.currentPosition == input.Length)

{

output.Text = "The input is valid.";

}

else

{

output.Text = "The input is invalid.";

}

}

}

}

**LL1 Parser:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace \_032\_MidLabQ2

{

public class LL1Parser

{

private readonly string input;

public int currentPosition;

public LL1Parser(string input)

{

this.input = input;

currentPosition = 0;

}

public void Parse()

{

S();

}

private void S()

{

E();

Match('$');

}

private void E()

{

T();

EPrime();

}

private void EPrime()

{

if (Match('+'))

{

T();

EPrime();

}

}

private void T()

{

F();

TPrime();

}

private void TPrime()

{

if (Match('\*'))

{

F();

TPrime();

}

}

private void F()

{

if (Match('('))

{

E();

Match(')');

}

else

{

Match('i');

Match('d');

}

}

private bool Match(char expected)

{

if (currentPosition >= input.Length)

{

return false;

}

if (input[currentPosition] == expected)

{

currentPosition++;

return true;

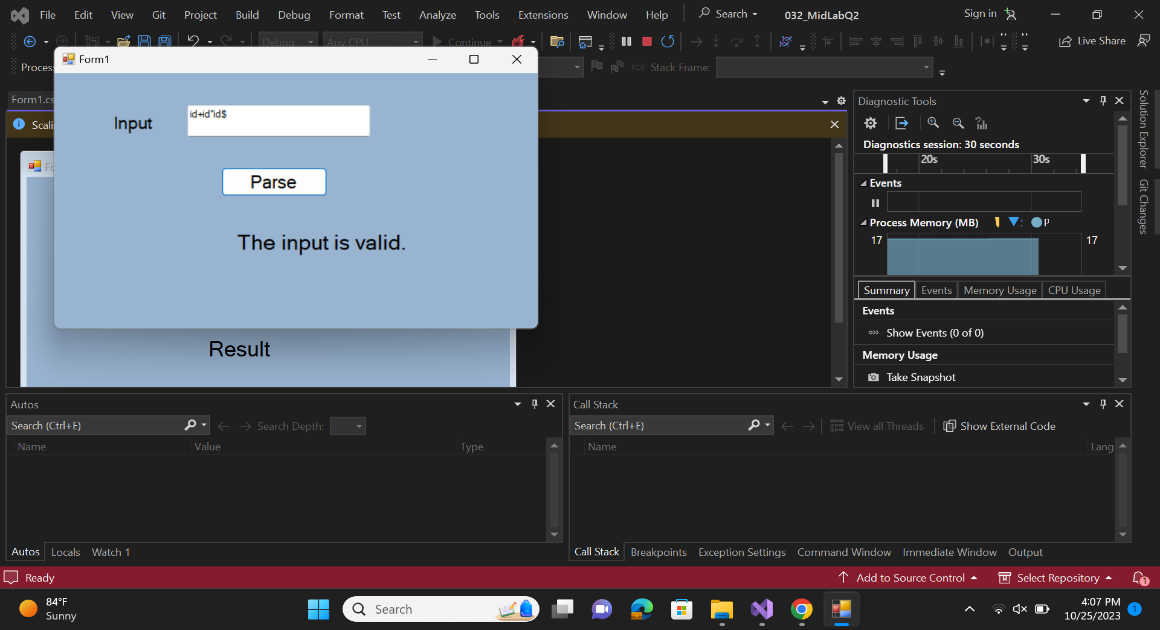
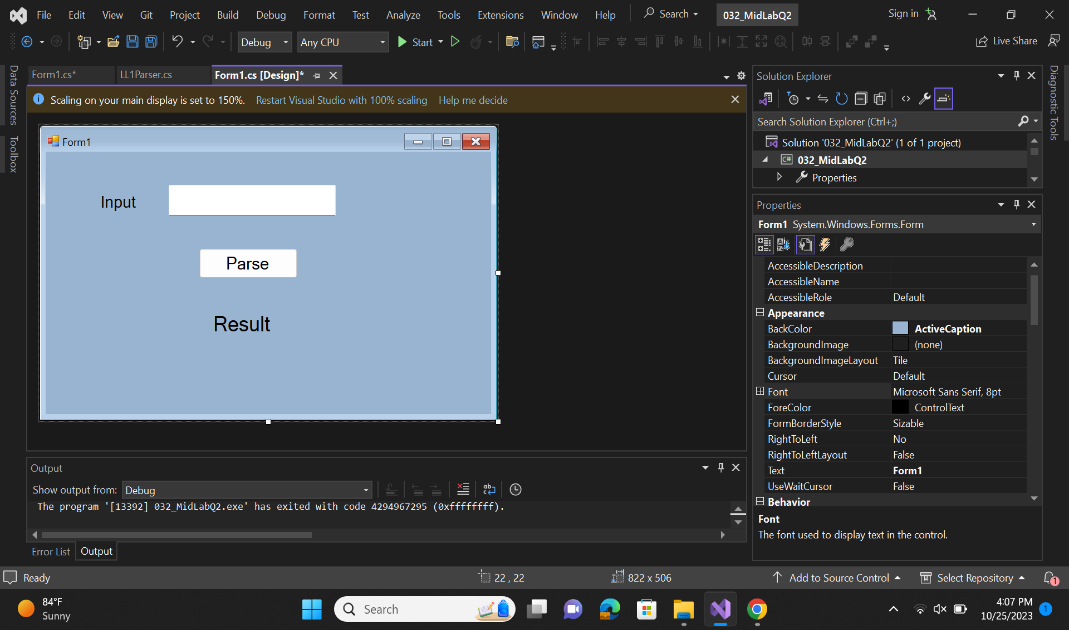
}

return false;

}

}

}



**Question No.3**

**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 \_032\_MidLab

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private string GeneratePassword()

{

Random random = new Random();

StringBuilder password = new StringBuilder();

// Rule (d): Must contain initials of first and last name, which are A and H

password.Append("AA");

// Rule (a): At least one uppercase alphabet

char uppercaseChar = (char)random.Next('A', 'Z' + 1);

password.Append(uppercaseChar);

// Rule (b): At least 4 numbers, 2 numbers must be 32

int[] requiredNumbers = { 3, 2 }; // These numbers represent "32"

int[] otherNumbers = { 0, 1, 4, 5, 6, 7, 8, 9 };

for (int i = 0; i < 2; i++)

{

int num = requiredNumbers[i];

password.Append(num);

}

for (int i = 0; i < 2; i++)

{

int num = otherNumbers[random.Next(otherNumbers.Length)];

password.Append(num);

}

// Rule (c): At least 2 special characters

string specialChars = "!@#$%^&\*()\_+";

for (int i = 0; i < 2; i++)

{

char specialChar = specialChars[random.Next(specialChars.Length)];

password.Append(specialChar);

}

// Shuffle the characters in the password for better randomness

string shuffledPassword = new string(password.ToString().ToCharArray().OrderBy(x => random.Next()).ToArray());

// Ensure the password length does not exceed 16 characters

return shuffledPassword.Length <= 16 ? shuffledPassword : shuffledPassword.Substring(0, 16);

}

private void button1\_Click(object sender, EventArgs e)

{

string password = GeneratePassword();

PasswordLabel.Text = password;

}

}

