Name: Muhammad Awais  
Reg #: FA20-BCS-020

CC Lab-Mid

Date: 25-Oct-23

Q. 1. Briefly describe the regex library of C#  
Answer: The regex library of C# is a set of classes and methods that provide support for regular expressions. Regular expressions are a powerful tool for searching, editing, and manipulating text. The C# regex library provides a comprehensive set of features for working with regular expressions, including:

* Matching and searching for patterns in text
* Extracting and replacing text
* Validating text input
* Generating text

The C# regex library is based on the .NET Framework's regular expression engine. This engine is a powerful and efficient implementation of the regular expression standard. The C# regex library provides a number of features that make it easy to use regular expressions in your applications, including:

* A comprehensive set of regular expression operators and metacharacters
* Support for named capture groups
* Support for compiling and caching regular expressions
* A variety of methods for matching and searching for patterns in text
* Methods for extracting and replacing text
* Methods for validating text input
* Methods for generating text

The C# regex library is a powerful tool for working with text. It is used in a wide variety of applications, including:

* Text editors
* Web browsers
* Compilers
* Databases
* Search engines
* Data mining applications

­

2. Make recursive descent or LL1 parser for the following grammer:

S -> E$

E -> T E'

E' -> + T E' | ε

T -> F T'

T' -> \* F T' | ε

F -> ( E ) | id

🡨 Code 🡪

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

string input = InputTextBox.Text;

Parser parser = new Parser(input);

parser.Parse();

// MessageBox.Show(parser.currentPosition + " ");

if (parser.currentPosition == input.Length)

{

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

}

else

{

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

}

}

}

public class Parser

{

private readonly string input;

public int currentPosition;

public Parser(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)

{

// MessageBox.Show(currentPosition + " " + input[currentPosition] + " " + expected);

if (currentPosition >= input.Length)

{

return false;

}

if (input[currentPosition] == expected)

{

currentPosition++;

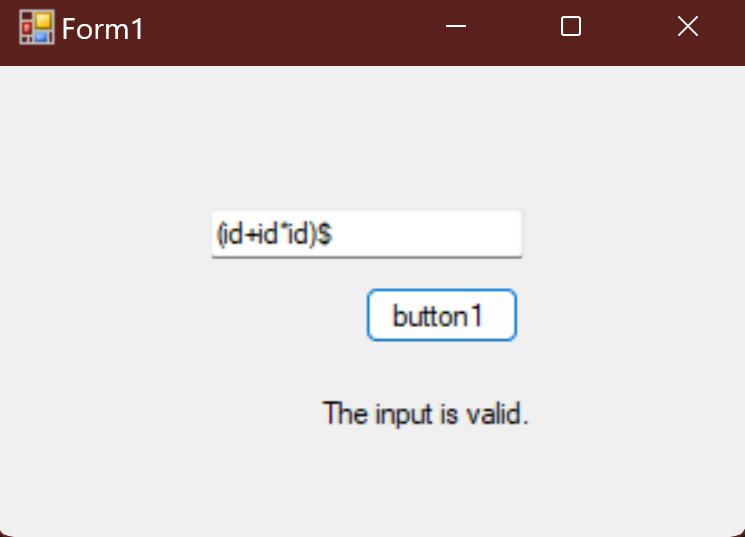
return true;

}

return false;

}

}



3. Make a Password generator according the following rules:

(a) Atleast one uppercase alphabet

(b) Atleast 4 numbers (where 2 numbers are ur registration number)

(c) Atleast 2 special characters

(d) Must contain initials of first and last name

(e) maximum length of 16

🡨 Code 🡪

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

string registrationNumber = RegistrationNumberTextBox.Text;

string firstName = FirstNameTextBox.Text;

string lastName = LastNameTextBox.Text;

string password = GeneratePassword(registrationNumber, firstName, lastName);

if (password != null)

{

GeneratedPasswordLabel.Text = password;

}

else

{

GeneratedPasswordLabel.Text = "Unable to generate a password that meets the criteria.";

}

}

private string GeneratePassword(string registrationNumber, string firstName, string lastName)

{

if (registrationNumber.Length < 2)

{

return null;

}

Random random = new Random();

// Registion number has 2 nums

int positionOfNum1 = random.Next(0, 4);

int positionOfNum2 = random.Next(0, 4);

while (positionOfNum1 == positionOfNum2)

{

positionOfNum2 = random.Next(0, 5);

}

string password = "";

int nums = random.Next(4, 8);

for (int i = 1; i <= nums; i++)

{

if (i == positionOfNum1)

{

password += registrationNumber[0];

}

else if (i == positionOfNum2)

{

password += registrationNumber[1];

}

else

{

password += random.Next(0, 10).ToString();

}

}

string specialCharacters = "!@#$%^&\*()\_-+=<>?";

int numberOfSpecialChars = random.Next(2, 5);

for (int i = 1; i <= numberOfSpecialChars; i++)

{

password += specialCharacters[random.Next(0, specialCharacters.Length)];

}

// Random upper case letter

int chars = random.Next(1, 4);

password += firstName[0];

for (int i = 1; i <= chars; i++)

{

password += Char.ToUpper((char)('a' + random.Next(26)));

}

password += lastName[0];

if (password.Length > 16)

{

password = password.Substring(0, 16);

}

return ShufflePassword(password);

}

static string ShufflePassword(string input)

{

char[] characters = input.ToCharArray();

Random random = new Random();

for (int i = characters.Length - 1; i > 0; i--)

{

int j = random.Next(0, i + 1);

char temp = characters[i];

characters[i] = characters[j];

characters[j] = temp;

}

return new string(characters);

}

}  
