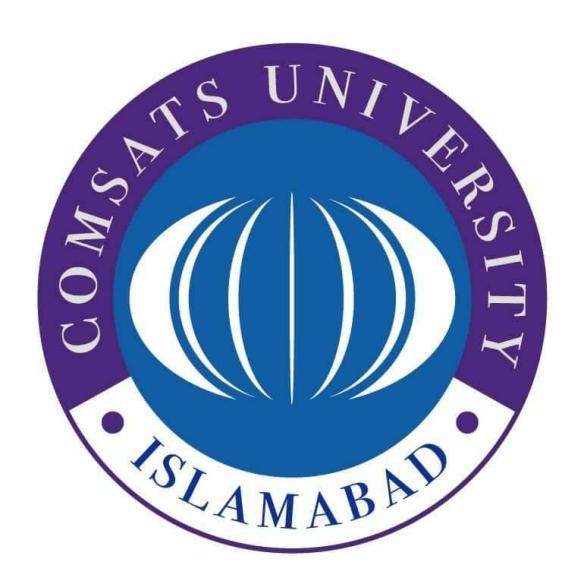
**MID-LAB**

****

**Name:M. ZOHAIB AHMER**

**Reg.no: FA20-BCS-038**

**Submitted to: DR. BILAL HAIDER BUKHARI**

**Dated:25/10/2023**

**Subject:Cc**

**COMSATS UNIVERSITY ISLAMABAD ATTOCK CAMPUS**

Q1:-Briefly describe the regex library of C# ?

Ans:-The C# regex library, which is based on the .NET Framework’s regular expression engine, is a powerful tool for working with text. It provides a simple syntax for creating regular expressions, which can be used to perform a wide variety of tasks such as validating input, parsing data, and transforming text.

Q2:-

ANS:-

using System;

class Parser

{

private string input;

private int index;

public Parser(string input)

{

this.input = input + "$"; // Append '$' to mark the end of input.

this.index = 0;

}

public bool Parse()

{

return ParseS();

}

private bool Match(char expected)

{

if (index < input.Length && input[index] == expected)

{

index++;

return true;

}

return false;

}

private bool ParseS()

{

// S -> E $

if (ParseE() && Match('$'))

{

return true;

}

return false;

}

private bool ParseE()

{

// E -> T E'

if (ParseT() && ParseEPrime())

{

return true;

}

return false;

}

private bool ParseEPrime()

{

// E' -> + T E' | ε

if (Match('+'))

{

if (ParseT() && ParseEPrime())

{

return true;

}

}

return true; // ε production

}

private bool ParseT()

{

// T -> F T'

if (ParseF() && ParseTPrime())

{

return true;

}

return false;

}

private bool ParseTPrime()

{

// T' -> \* F T' | ε

if (Match('\*'))

{

if (ParseF() && ParseTPrime())

{

return true;

}

}

return true; // ε production

}

private bool ParseF()

{

// F -> ( E ) | id

if (Match('('))

{

if (ParseE() && Match(')'))

{

return true;

}

}

else if (input.Substring(index, 2) == "id") // Check for 'id'

{

index += 2; // Move the index by 2 to skip 'id'

return true;

}

return false;

}

}

class Program

{

static void Main(string[] args)

{

string input = "id + id \* id $"; // Replace with your input string

Parser parser = new Parser(input);

if (parser.Parse())

{

Console.WriteLine("Input is valid according to the grammar.");

}

else

{

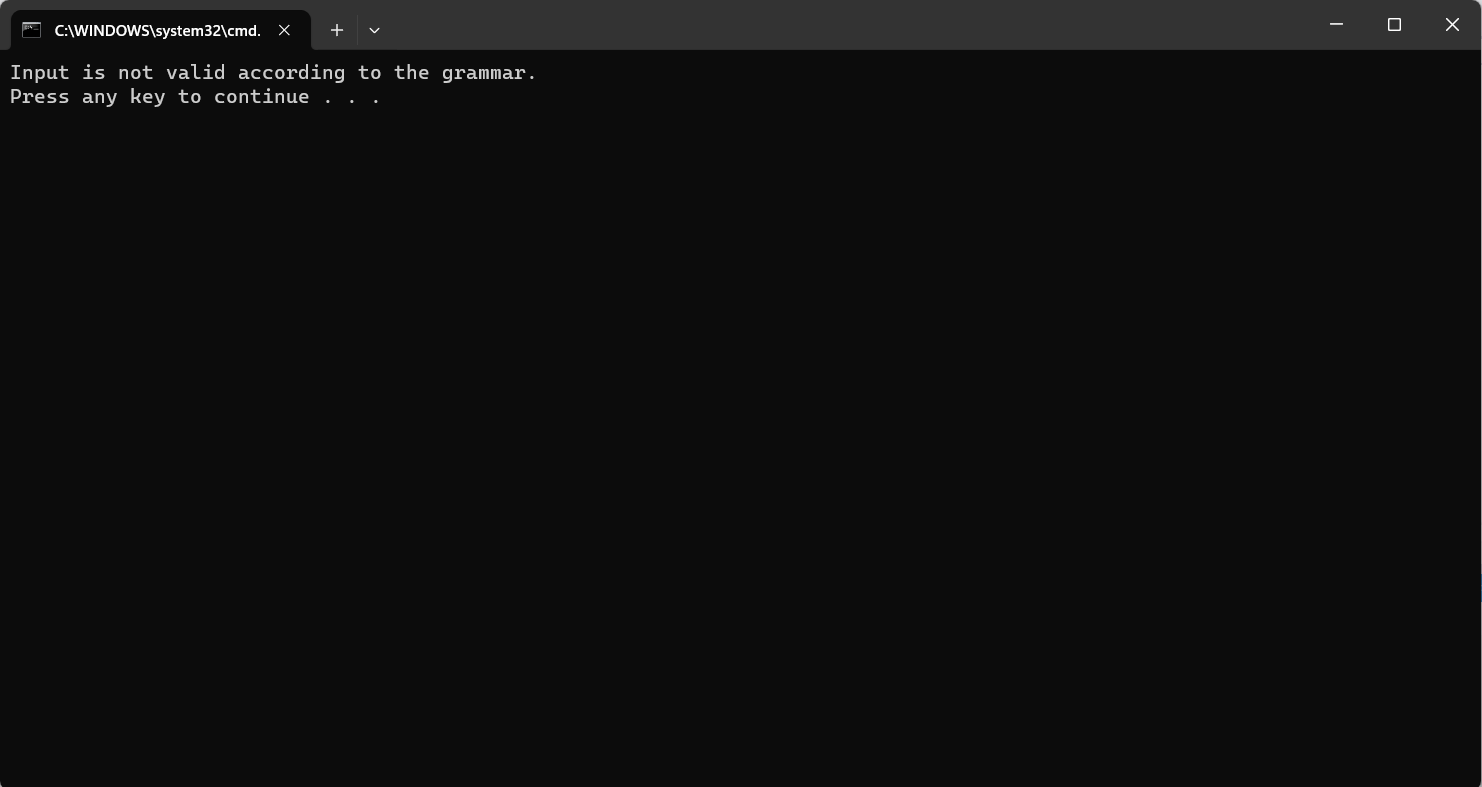
Console.WriteLine("Input is not valid according to the grammar.");

}

}

}

OUTPUT:-



Q3:- Make a Password generator according the following rules:

(a) Atleast one uppercase alphabet

(b) Atleast 4 numbers

(c) Atleast 2 special characters

(d) Must contain initials of first and last name

(e) maximum length of 16?

ANS:-

using System;

using System.Text;

using System.Linq;

class PasswordGenerator

{

static void Main(string[] args)

{

string firstName = "zohaib"; // Replace with your first name

string lastName = "ahmer"; // Replace with your last name

string password = GeneratePassword(firstName, lastName);

Console.WriteLine("Generated Password: " + password);

}

static string GeneratePassword(string firstName, string lastName)

{

StringBuilder password = new StringBuilder();

// Rule (d) Must contain initials of first and last name

password.Append(char.ToLower(firstName[0]));

password.Append(char.ToLower(lastName[0]));

// Rule (a) At least one uppercase alphabet

char uppercaseLetter = (char)('A' + new Random().Next(26));

password.Append(uppercaseLetter);

// Rule (b) At least 4 numbers

for (int i = 0; i < 4; i++)

{

int randomNumber = new Random().Next(10);

password.Append(randomNumber);

}

// Rule (c) At least 2 special characters

string specialCharacters = "!@#$%^&\*()-\_=+[]{}|;:',.<>/?";

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

{

char specialChar = specialCharacters[new Random().Next(specialCharacters.Length)];

password.Append(specialChar);

}

// Ensure the password length is at most 16 characters

string generatedPassword = password.ToString();

if (generatedPassword.Length > 16)

{

generatedPassword = generatedPassword.Substring(0, 16);

}

return generatedPassword;

}

}

OUTPUT:-

