

**COMPILER CONSTRUCTION**

**LAB MID**

**NAME:**

**Muhammad Aamir Bin Habib**

**REG. NO:**

**FA20-BCS-010**

**CLASS:**

**BCS-7B**

**Date:**

**25-October-2023**

**QUESTION NO.1**

Regular Expressions (Regex) are a powerful and versatile tool in computer science used for pattern matching within strings of text. They consist of a sequence of characters that define a search pattern. These patterns can be used to perform a wide range of operations on text data, including searching for specific substrings, validating data, extracting information, and manipulating text.

Regular expressions are composed of ordinary characters (like letters and digits) and special characters (met characters) that have special meanings, allowing you to define complex patterns to match against text.

• PURPOSES OF REGULAR EXPRESSIONS:

1. PETTERN MATCHING:

Regular expressions are primarily used to match and search for specific patterns or sequences of characters within a given text. This is helpful for tasks like finding email addresses, URLs, or specific keywords in a document.

2. DATA VALIDATION:

They are widely employed for validating user input or data to ensure it conforms to a particular format. For example, validating phone numbers, ZIP codes, or credit card numbers.

3. TEXT EXTRACTION:

Regular expressions can extract specific information from text data, such as extracting dates, numbers, or other structured data from a larger document.

4. TEXT TRANSFORMATION:

They are used for text manipulation, such as replacing or reformatting text to meet specific requirements. This can be used to sanitize and format data.

• EXAMPLES:

Suppose you have a string containing multiple email addresses, and you want to extract them using a regex in C#:

using System;

using System.Text.RegularExpressions;

string input = "Emails: usama@gmail.com, maan@gmail.com, and minhas@gmail.com";

string pattern = @"\w+@\w+\.\w+";

Regex regex = new Regex(pattern);

MatchCollection matches = regex.Matches(input);

foreach (Match match in matches)

{

Console.WriteLine("Email: " + match.Value);

}

**QUESTION NO.2 & 3**

* **CODE:**

**Class (Form.cs):**

using System;

using System.Text;

using System.Windows.Forms;

using labMID\_010; // Adjust the namespace if needed

namespace labMID\_010

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

string input = textBox1.Text;

LL1Parser parser = new LL1Parser(input);

parser.Parse();

if (parser.currentPosition == input.Length)

{

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

}

else

{

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

}

}

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

{

string password = GeneratePassword();

passwordOutput.Text = password;

}

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("AH");

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

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

password.Append(uppercaseChar);

// Rule (b): At least 4 numbers, two numbers must be 1 and 0

int[] requiredNumbers = { 1, 0 };

int[] otherNumbers = { 2, 3, 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

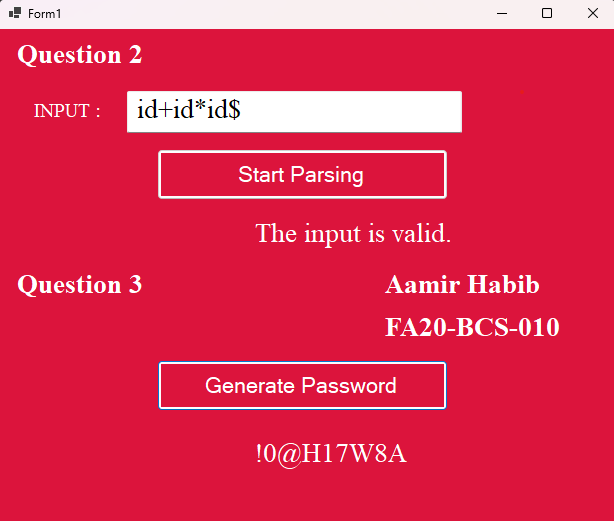
return shuffledPassword.Length <= 16 ? shuffledPassword : shuffledPassword.Substring(0, 16);

}

}

}

**OUTPUT:**

****

**Class (LL1Parser.cs):**

using System;

namespace labMID\_010

{

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;

}

}

}

**OUTPUT:**

