

COMPILER CONSTRUCTION

NAME:

USAMA MANSOOR

REG. NO:

FA20-BCS-026

CLASS:

BCS-7B

SUBMITTED TO:

MR BILAL HAIDER

DATE:

25-10-2023

- **QUESTION NO 01:**

Briefly describe the regex library of C#

- **ANSWER:**

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. PATTERN 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 02:**
- **Recursive descent parser**
- **CODE:**

Creating new class inside the project named (LL1Parser.cs)

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;
```

```
namespace Lab_Mid_Recurrsive_Descent_Parser
```

```
{
```

```
    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;
}
}
}

```

Form.cs 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 Lab_Mid_Recurrive_Descent_Parser
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}

```

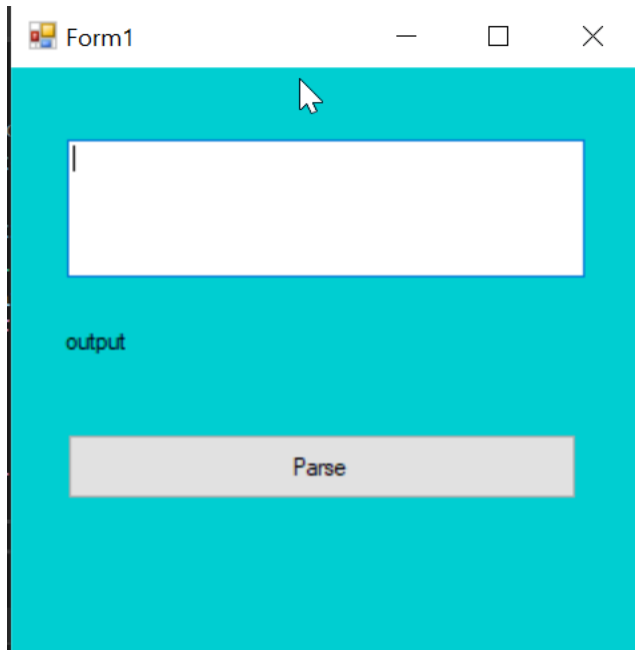
```
private void parse_Click(object sender, EventArgs e)
{
    string input = inputBox.Text;

    LL1Parser parser = new LL1Parser(input);

    parser.Parse();

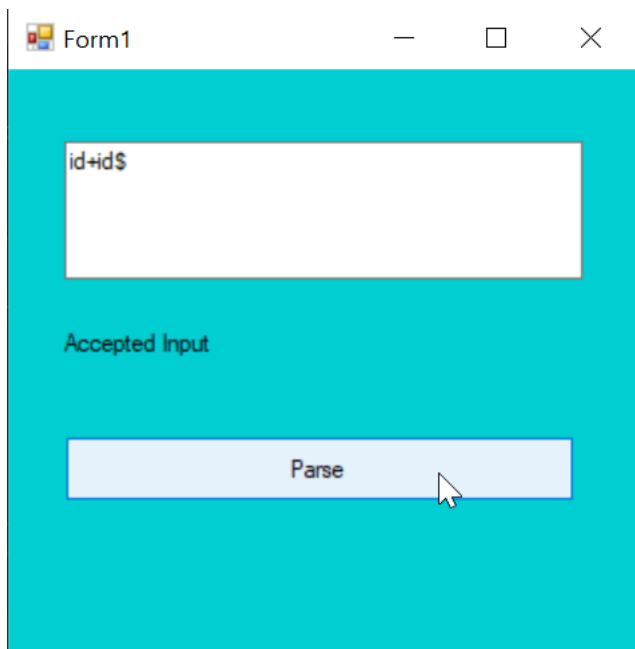
    if (parser.currentPosition == input.Length)
    {
        outputLabel.Text = "The input is valid.";
    }
    else
    {
        outputLabel.Text = "The input is invalid.";
    }
}
}
```


- **OUTPUT:**
- **GENERAL OUTPUT:**



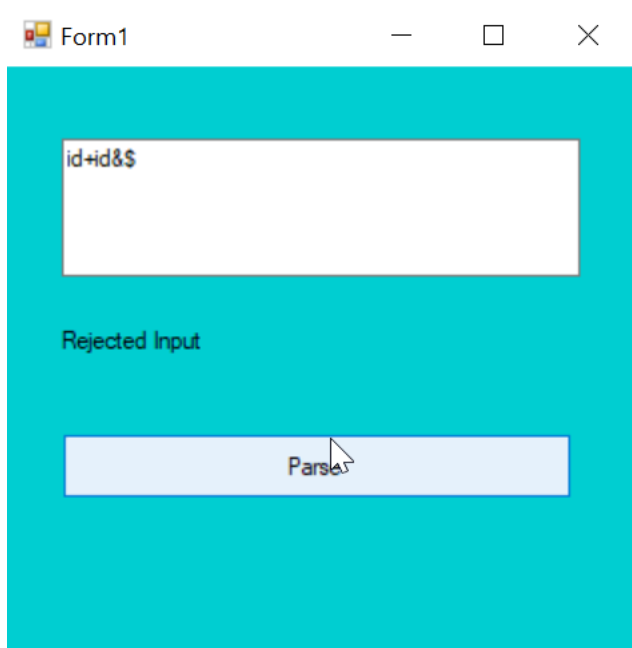
A screenshot of a Windows application window titled "Form1". The window has a cyan background and standard Windows window controls (minimize, maximize, close). Inside the window, there is a text input field at the top. Below the input field is the label "output". At the bottom of the window is a grey button labeled "Parse". A mouse cursor is visible near the top of the input field.

ACCEPTED INPUT:



A screenshot of the same "Form1" window. The text input field now contains the text "id+id\$". Below the input field is the label "Accepted Input". The "Parse" button is now highlighted in light blue, and a mouse cursor is pointing at it.

REJECTED INPUT:



- **QUESTION NO 03:**
- **PASSWORD GENERATOR:**
- **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 Lab_Mid_Password_Gen
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void generate_Click(object sender, EventArgs e)
        {
            string password = GeneratePassword();

            output.Text = password;

            int count = 0;

            for (int i = 0; i < password.Length; i++)
            {
                count = count + 1;
            }

            length.Text = count.ToString();
        }

        private string GeneratePassword()
        {
            Random random = new Random();

            StringBuilder password = new StringBuilder();

```

```
password.Append("UM");
```

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

```
password.Append(uppercaseChar);
```

```
int[] requiredNumbers = { 2, 6 };
```

```
int[] otherNumbers = { 0, 1, 3, 4, 5, 7, 8, 9 };
```

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

```
{
```

```
    int num = requiredNumbers[i];
```

```
    password.Append(num);
```

```
}
```

```
for (int i = 0; i < 3; i++)
```

```
{
```

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

```
    password.Append(num);
```

```
}
```

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

```
int chrr = random.Next(2, 7);
```

```
for (int i = 0; i < chrr; i++)
```

```

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

    password.Append(specialChar);
}

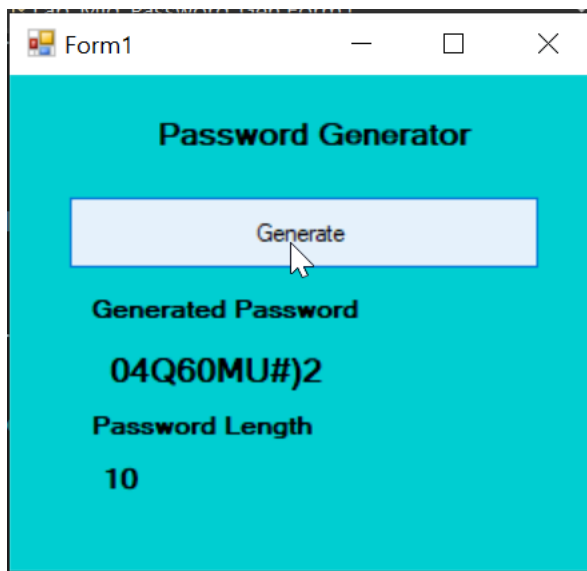
string shuffledPassword = new string(password.ToString().ToCharArray().OrderBy(x =>
random.Next()).ToArray());

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

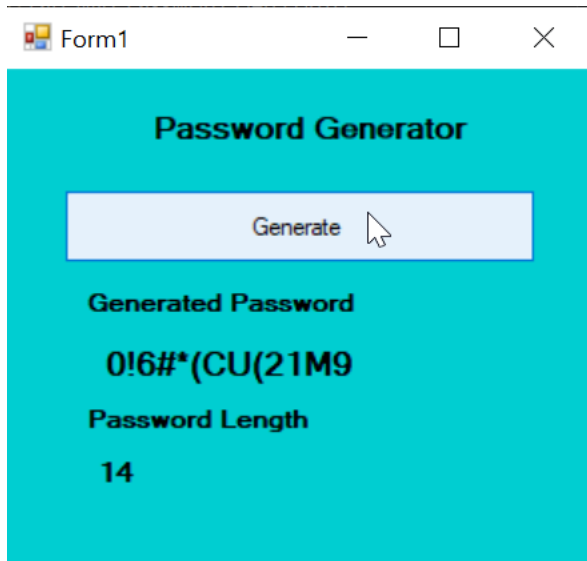
```

- **OUTPUT:**

OUTPUT 1:

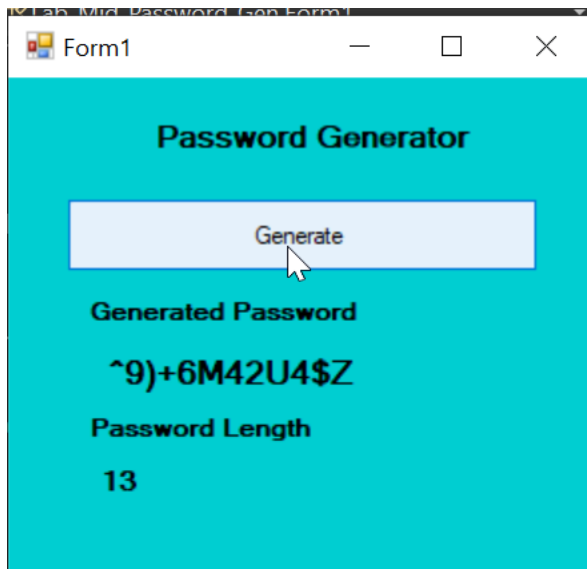


OUTPUT 2:



The screenshot shows a window titled "Form1" with a cyan background. At the top, it says "Password Generator". Below that is a light blue button labeled "Generate" with a mouse cursor hovering over it. Under the button, the text "Generated Password" is followed by the password "0!6#*(CU(21M9". Below that, the text "Password Length" is followed by the number "14".

OUTPUT 3:



The screenshot shows a window titled "Form1" with a cyan background. At the top, it says "Password Generator". Below that is a light blue button labeled "Generate" with a mouse cursor hovering over it. Under the button, the text "Generated Password" is followed by the password "^9)+6M42U4\$Z". Below that, the text "Password Length" is followed by the number "13".