

MUHAMMAD MUAAB SHOAIB

FA20-BCS-074

CC MID LAB

25-OCT-2023

Q1: Briefly describe the regex library of C#.

The regex library of C# is a powerful tool for working with regular expressions. It provides a set of classes and methods that allow you to match, search, and replace text based on regular expression patterns.

The most important class in the regex library is the `Regex` class. This class represents a regular expression and provides a variety of methods for working with it. For example, you can use the **`Match()`** method to find a match for the regular expression in a given string, and the `Replace()` method to replace all matches for the regular expression with a new string.

The regex library also includes several other classes and methods that can be useful for working with regular expressions. For example, the **`MatchCollection()`** class represents a collection of matches for a regular expression, and the `Group` class represents a group of characters within a match.

Key Classes in C# Regex Library:

1. **Regex:** The `Regex` class is the primary class in the library. It represents a compiled regular expression pattern and provides methods for pattern matching and replacement.
2. **Match:** The `Match` class represents a single match of a regular expression pattern in an input string. It provides information about the matched text and its position.
3. **MatchCollection:** This class represents a collection of `Match` objects. It is returned by methods like `Regex.Matches()` when you want to find all matches in an input string.

Basic Operations:

1. **Pattern Matching (Regex.Match):** You can use `Regex.Match()` to find the first occurrence of a regular expression pattern in an input string. It returns a `Match` object containing information about the first match.
2. **Pattern Matching (Regex.Matches):** The `Regex.Matches()` method finds all occurrences of a pattern in an input string and returns a `MatchCollection` containing all the matches.
3. **Pattern Replacement (Regex.Replace):** You can use `Regex.Replace()` to replace all occurrences of a pattern in an input string with a specified replacement string.

Common Regex Elements:

1. **Literals:** Characters that match themselves, e.g., "abc" matches the string "abc."
2. **Character Classes:** Square brackets define character classes, like `[A-Za-z]` to match any uppercase or lowercase letter.
3. **Quantifiers:** Specify how many times a character or group should appear. For example, `*` matches zero or more times, and `+` matches one or more times.
4. **Anchors:** `^` matches the start of a line, and `\$` matches the end.
5. **Escape Sequences:** Backslashes `\` are used to escape special characters. For example, `\. ` matches a literal period, and `\d` matches a digit.

Here is an example of how to use the regex library to match and replace text:

C#

```
// Create a regular expression object
Regex regex = new Regex(@"\d+");

// Match the regular expression in the input string
Match match = regex.Match("This string contains 123 numbers");

// If there is a match, replace it with the string "numbers"
if (match.Success)
{
    string output = match.Result.Replace("123 numbers", "numbers");

    Console.WriteLine(output); // This string contains numbers
}
```

The regex library is a very powerful tool for working with text, and it can be used to solve a wide variety of problems.

Here are some of the benefits of using the regex library of C#:

- It is very efficient and can be used to process large amounts of text quickly.
- It is very flexible and can be used to create complex regular expression patterns.
- It is well-documented and there are many resources available to help you learn how to use it.

If you need to work with text in your C# applications, I highly recommend using the regex library.

Q2: Make recursive descent or LL1 parser for the following grammar:

$S \rightarrow E\$$

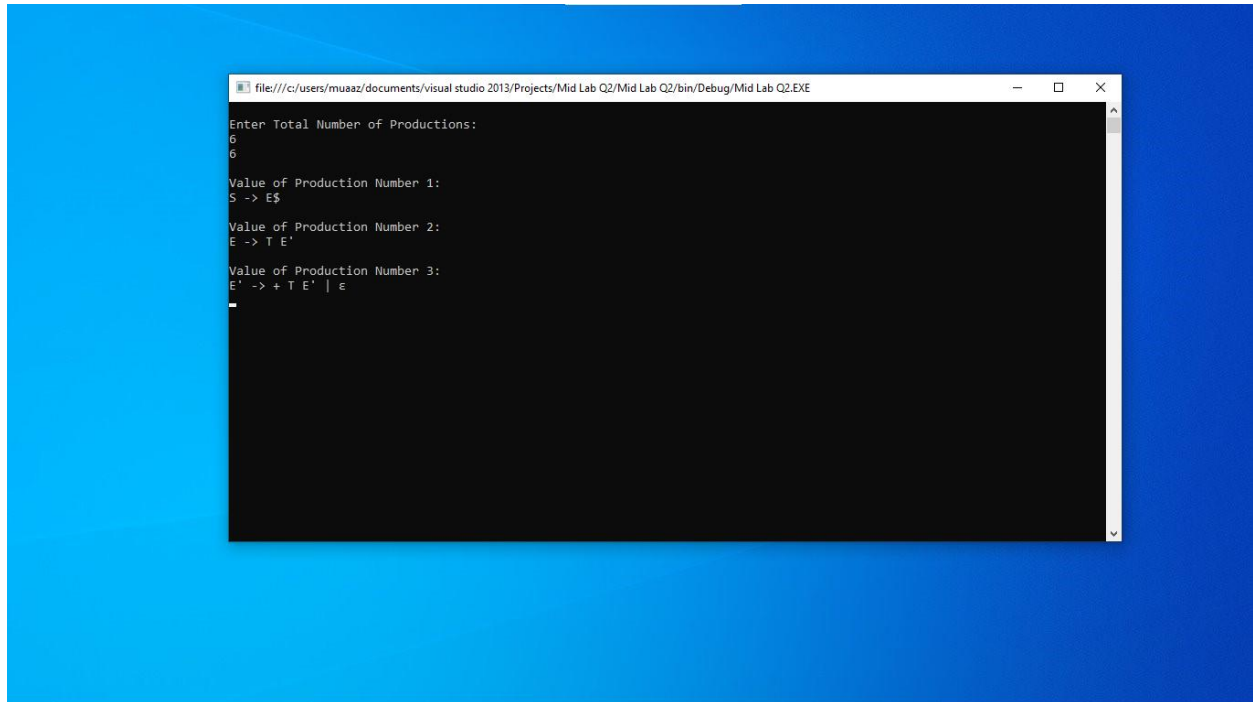
$E \rightarrow T E'$

$E' \rightarrow + T E' \mid \epsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' \mid \epsilon$

$F \rightarrow (E) \mid id$



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Mid_Lab_Q2
{
    class Program
    {
        static int limit, x = 0;

        static char[,] production = new char[10, 10];
        static char[] array = new char[10];
        static void Main(string[] args)
        {
            for (int i = 0; i < 10; i++)
            {
                for (int j = 0; j < 10; j++)
                {
                    //To signify empty space.
                    production[i, j] = '-';
                }
            }
            int count = 0;
            char option, ch;
            Console.WriteLine("\nEnter Total Number of Productions:\t");
```

```

limit = Convert.ToInt32(Console.ReadLine());
Console.WriteLine(limit);
for (count = 0; count < limit; count++)
{
    Console.WriteLine("\nValue of Production Number {0}:\t", count + 1);
    String temp = Console.ReadLine();
    for (int i = 0; i < temp.Length; i++)
    {
        production[count, i] = temp[i];
    }
}
// Keep asking the user for non-terminal for which follow_set is needed.
do
{
    x = 0;
    Console.WriteLine("\nEnter production Value to Find Follow:\t");

    ch = Console.ReadKey().KeyChar;
    find_follow(ch);
    Console.WriteLine("\nFollow Value of {0}:\t", ch);
    for (count = 0; count < x; count++)
    {
        Console.Write(array[count]);
    }
    Console.Write("}\n");
    Console.Write("To Continue, Press Y:\t");
    option = ch = Console.ReadKey().KeyChar;
} while (option == 'y' || option == 'Y');
for (int i = 0; i < 10; i++)
{
    for (int j = 0; j < 10; j++)
    {
        Console.Write(production[i, j]);
    }
    Console.Write("\n");
}
Console.ReadKey();
}
static void find_follow(char ch)
{
    int i = 0, j;
    for (int k = 0; k < 10; k++)
    {
    }
    int length = 10;
    if (Convert.ToChar(production[0, 0]).Equals(ch))
    {
        array_manipulation('$');
    }
    for (i = 0; i < limit; i++)
    {
        for (j = 2; j < length; j++)
        {
            if (Convert.ToChar(production[i, j]).Equals(ch))
            {

```

```

        if (Convert.ToChar(production[i, j + 1]).Equals('\0'))
        {
            find_first(Convert.ToChar(production[i, j + 1]));
        }
        if (Convert.ToChar(production[i, j + 1]).Equals('\0') && ch.Equals(Convert.ToChar(production[i, 0])))
        {
            find_follow(Convert.ToChar(production[i, 0]));
        }
    }
}

static void find_first(char ch)
{
    int i = 0, k;
    //Check for uppercase letter.
    int val = System.Convert.ToInt32(ch);
    if (!(val >= 97 && val <= 122))
    {
        array_manipulation(ch);
    }
    for (k = 0; k < limit; k++)
    {
        if (production[k, 0].Equals(ch))
        {
            if (production[k, 2].Equals('$'))
            {
                find_follow(Convert.ToChar(production[i, 0]));
            }
            //Check for lowercase.
            else if (Convert.ToInt32((production[k, 2])) >= 97 && Convert.ToInt32((production[k, 2])) <= 122)
            {
                array_manipulation(Convert.ToChar(production[k, 2]));
            }
            else
            {
                find_first(Convert.ToChar(production[k, 2]));
            }
        }
    }
}

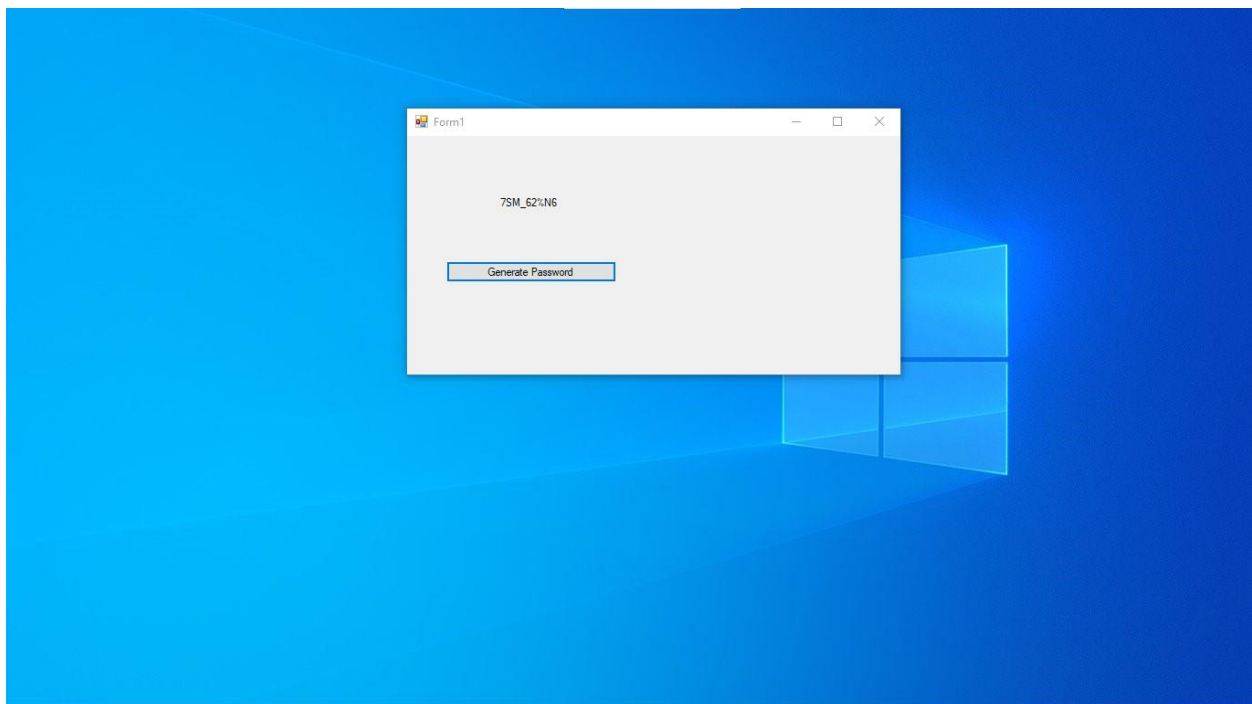
static void array_manipulation(char ch)
{
    int count;
    for (count = 0; count <= x; count++)
    {
        if (array[count].Equals(ch))
        {
            return;
        }
    }
    array[x++] = ch;
}
}

```

}

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

- (a) At least one uppercase alphabet**
- (b) At least 4 numbers**
- (c) At least 2 special characters**
- (d) Must contain initials of first and last name**
- (e) maximum length of 16**



```
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;  
using System.Text;
```

```

using System.Text.RegularExpressions;

namespace Mid_Lab_Q3
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
        }
        private void button1_Click(object sender, EventArgs e)
        {
            string firstName = "Muaaz";
            string lastName = "Shoaib";

            if (firstName.Length < 1 || lastName.Length < 1)
            {
                MessageBox.Show("Please enter your first and last name.");
                return;
            }

            // Create a StringBuilder to build the password
            StringBuilder password = new StringBuilder();

            // Add initials of first and last name
            password.Append(firstName[0]);
            password.Append(lastName[0]);

            // Generate random uppercase alphabet
            Random random = new Random();
            password.Append((char)random.Next('A', 'Z' + 1));

            // Generate 4 random numbers
            for (int i = 0; i < 4; i++)
            {
                password.Append((char)random.Next('0', '9' + 1));
            }

            // Generate 2 special characters
            string specialCharacters = "!@#$%^&*()_-=<>?";
            for (int i = 0; i < 2; i++)
            {
                password.Append(specialCharacters[random.Next(specialCharacters.Length)]);
            }

            // Shuffle the password characters for better security
            password = ShuffleString(password);

            // Limit the password to a maximum length of 16
            if (password.Length > 16)
            {
                password.Length = 16;
            }
        }
    }
}

```

```
        // Display the generated password
        label1.Text = password.ToString();
    }
    private StringBuilder ShuffleString(StringBuilder str)
    {
        Random random = new Random();
        int n = str.Length;
        while (n > 1)
        {
            n--;
            int k = random.Next(n + 1);
            char value = str[k];
            str[k] = str[n];
            str[n] = value;
        }
        return str;
    }
}
}
```