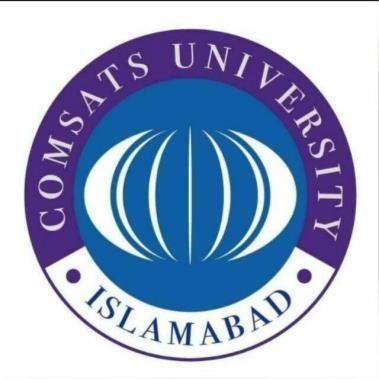
**Comsats university Islamabad, Attock Campus**

**Department of Computer Science**

**Lab Mid**



**Submitted By:**

**Sofia Ahmed(FA20-BCS-053)**

**Subject: Compiler Construction**

**Course: BCS-7A**

**Question 1**

**Describe functioning of regex C# library , give examples of patterns,seperators and anchors e.t.c.**

Regular expressions (regex) in C# are supported through the System.Text.RegularExpressions namespace, which provides classes and methods for working with regular expressions. Here, I'll describe the basic functioning of the C# regex library and provide examples of common regex patterns, separators, anchors, and more.

1. **Namespace and Classes:** The System.Text.RegularExpressions namespace contains the key classes for working with regular expressions in C#:

**Regex:** The Regex class represents a compiled regular expression pattern. It provides methods for matching and manipulating text using regular expressions.

**Match:** The Match class represents the results of a single regular expression match. It contains information about the matched text, groups, and more.

1. **Creating a Regex Pattern:**

To create a regular expression pattern, you can use the Regex class constructor.

For example:

using System.Text.RegularExpressions;

string pattern = @"[0-9]+"; // Matches one or more digits

Regex regex = new Regex(pattern);

1. **Common Regex Elements:**

**Literals:** Characters like letters and digits match themselves. For example, a matches the letter 'a', and 123 matches the digits '123'.

**Character Classes:** Use square brackets to specify a set of characters. For example, [aeiou] matches any vowel.

**Quantifiers:** Specify the number of occurrences with symbols like \* (zero or more), + (one or more), ? (zero or one), and {n} (exactly n times).

**Anchors:**

^: Matches the start of a line.

$: Matches the end of a line.

\b: Matches a word boundary.

**Escape Sequences:** Special characters like . or \* need to be escaped with a backslash (e.g., \.).

1. **Matching Text:**

The Regex.Match() method is used to find the first occurrence of a pattern in a string.

string text = "The quick brown fox jumps over the lazy dog.";

Match match = regex.Match(text);

if (match.Success)

{

string matchedText = match.Value;

}

1. **Matching Multiple Occurrences:**

To find all occurrences of a pattern in a string, you can use the Regex.Matches() method:

MatchCollection matches = regex.Matches(text);

foreach (Match match in matches)

{

string matchedText = match.Value;

}

1. **Replacing Text:**

The Regex.Replace() method can be used to replace matched patterns in a string:

string replacedText = regex.Replace(text, "\*\*\*\*");

1. **Examples:**

Matching email addresses:

string emailPattern = @"[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}";

**Extracting phone numbers:**

string phonePattern = @"\b\d{3}-\d{3}-\d{4}\b";

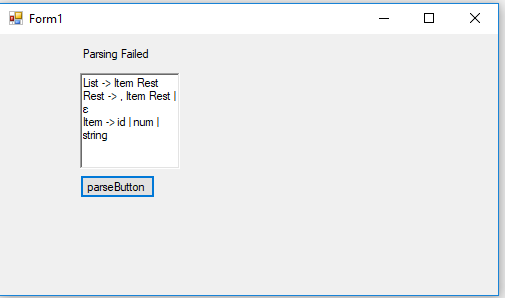
**Matching URLs:**

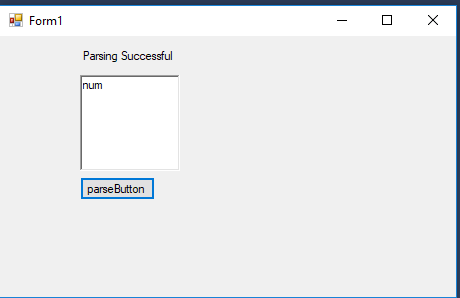
string urlPattern = @"https?://\S+";

These are just some basic examples. Regular expressions can get quite complex, and you can build intricate patterns to suit your specific needs. Remember to use appropriate anchors, quantifiers, and character classes to fine-tune your regex patterns for accurate matching.

**Question 2:**

**Output:**





**Question 3:**

**Output:**

