

**NAME : HASNAIN AHMED**

**REGISTER NO : FA20-BCS-005**

**DATE : 26-OCT-2023**

**COMPILER CONSTRUCTION MID LAB**

**Question 1:**

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

Here's an overview of how the C# Regex library works, along with examples of patterns, separators, anchors, and more.

* **Import the Namespace:**

Before using regular expressions in C#, make sure to import the System.Text.RegularExpressions namespace:

**using System.Text.RegularExpressions;**

**2. Create a Regex Pattern:**

A regex pattern is a sequence of characters that defines a search pattern. It can include various elements, such as literals, character classes, quantifiers, anchors, and more.

**string pattern = @"[0-9]{3}-[0-9]{2}-[0-9]{4}";**

In this example, the pattern matches a social security number (e.g., "123-45-6789")**.**

**3. Match Text:**

You can use the Regex.Match() method to find the first occurrence of the pattern in a string. For example:

**string text = "My SSN is 123-45-6789.";**

**Match match = Regex.Match(text, pattern);**

**if (match.Success)**

**{**

**Console.WriteLine("Match found: " + match.Value);**

**}**

**4. Matches and Groups:**

The Match object contains information about the matched text. You can access matched groups and their values:

**if (match.Success)**

**{**

**Console.WriteLine("Full match: " + match.Value);**

**foreach (Group group in match.Groups)**

**{**

**Console.WriteLine("Group " + group.Index + ": " + group.Value);**

**}**

**}**

**5. Regex Options:**

You can specify various options for your regex pattern. For example, RegexOptions.IgnoreCase makes the pattern case-insensitive:

**string pattern = @"apple";**

**string text = "I like Apple.";**

**Match match = Regex.Match(text, pattern, RegexOptions.IgnoreCase);**

**6. Common Elements in Regex Patterns:**

* Literals: Match characters literally (e.g., "apple" matches "apple").
* Character Classes: Use [ ] to define a set of characters (e.g., [0-9] matches any digit).
* Quantifiers: Specify repetition (e.g., \* for zero or more, + for one or more, ? for zero or one).
* Anchors: ^ for the start of a line, $ for the end of a line.
* Escape Sequences: Use \ to escape special characters (e.g., \. matches a literal period).

7. **Regex Escape:**

To match a specific character that's also a regex metacharacter, use Regex.Escape():

**string pattern = Regex.Escape("www.example.com");**

**8. Splitting Strings:**

You can use Regex.Split() to split a string based on a regex pattern:

**string text = "apple,banana,carrot";**

**string[] items = Regex.Split(text, ",");**

**9. Replacing Text:**

Use Regex.Replace() to replace text based on a pattern:

**string input = "I have a cat and a dog.";**

**string pattern = "cat|dog";**

**string replacement = "pet";**

**string result = Regex.Replace(input, pattern, replacement);**

**10. Backreferences:**

You can capture groups and reference them in the replacement pattern**:**

**string input = "apple,orange,banana";**

**string pattern = @"(\w+),\1";**

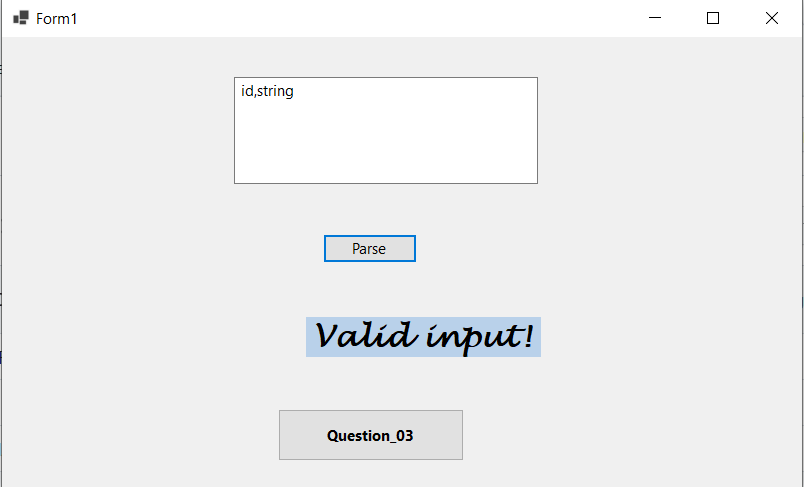
**string replacement = "$1"; // Replaces "banana,banana" with just "banana"**

**string result = Regex.Replace(input, pattern, replacement);**

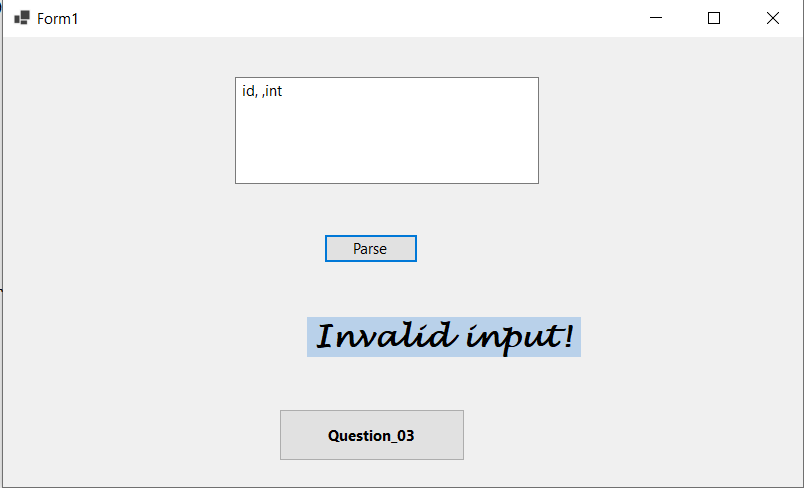
Regex in C# is a powerful tool for text manipulation, allowing you to search, match, split, and replace text based on complex patterns. Be aware that regex patterns can become quite intricate, and understanding the various metacharacters and techniques is crucial for effective use.

**Question 02:**

**Input : id,string**

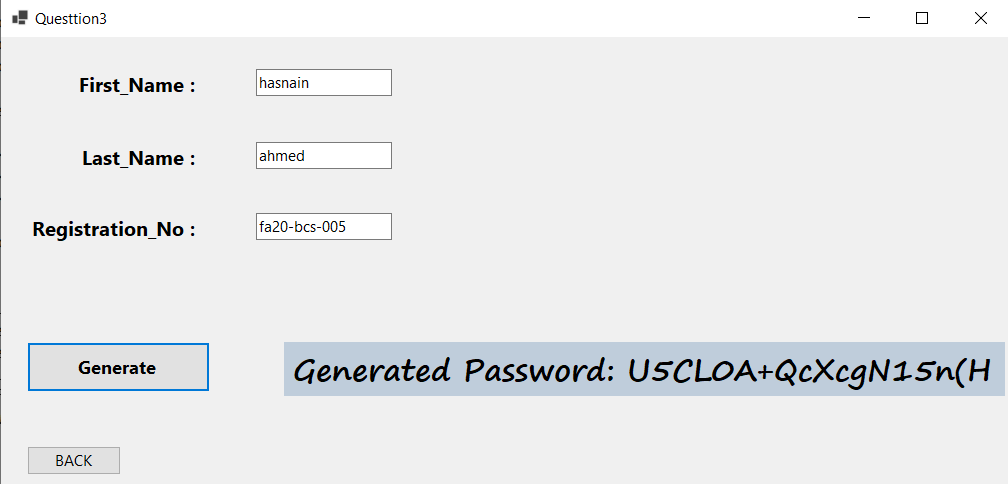


**Input : id, ,int**



**Question 03:**

**Clicked on gernate button one time:**



**Clicked on gernate button second time:**

