Regular Expressions in C#

In C#, Regular Expression is a pattern which is used to parse and check whether the given input text is matching with the given pattern or not.

The .Net Framework provides a regular expression engine that allows the pattern matching.

Patterns may consist of any character literals, operators or constructors.

The Regex Class

C# provides a class termed as Regex which can be found in System.Text.RegularExpression namespace.

This class will perform two things:

- Parsing the inputting text for the regular expression pattern.
- Identify the regular expression pattern in the given text.

We need to create an instance of the Regex class:

```
Regex regex = new Regex(pattern);
```

pattern - It may consist of any character literals, operators or constructors.

The Regex class provides a IsMatch() method which returns True if the string that we pass matches the regex pattern.

Example 1: To matches a single character in the list [abc]

```
using System;
using System.Text.RegularExpressions;
class Program
{
    static void Main(string[] args)
    {
        string pattern = "[abc]";
        Regex regex = new Regex(pattern);

        if (regex.IsMatch("car"))
            Console.WriteLine("Matched");
        else
            Console.WriteLine("Not Matched");
        if (regex.IsMatch("peek"))
            Console.WriteLine("Matched");
        else
            Console.WriteLine("Not Matched");

        Console.ReadKey();
    }
}
```

```
Output:
Matched
Not Matched
```

In this context, the pattern [abc] identifies a single character within a given input string that can be either 'a,' 'b,' or 'c.' For instance, if the input string is 'car,' it successfully finds a match because it contains either 'c' or 'a.' Conversely, in a different scenario, like the input string 'peek,' there is no match because it lacks any occurrence of 'a,' 'b,' or 'c.'

Example 2: Program to match the given input is any English alphabetic letters both uppercase and lowercase.

```
using System;
using System.Text.RegularExpressions;
class Program
{
    static void Main(string[] args)
    {
        string pattern = "[a-zA-Z]";
        Regex regex = new Regex(pattern);

        Console.WriteLine(regex.IsMatch("hello"));
        Console.WriteLine(regex.IsMatch("HELLO"));
        Console.WriteLine(regex.IsMatch("1234"));

        Console.ReadKey();
    }
}
```

```
Output:
True
True
False
```

Example 3: Program to match the given input is any digit

```
using System;
using System.Text.RegularExpressions;
class Program
{
    static void Main(string[] args)
    {
        string pattern = "[0-9]";
        Regex regex = new Regex(pattern);

        Console.WriteLine(regex.IsMatch("hello"));
        Console.WriteLine(regex.IsMatch("1234"));

        Console.ReadKey();
    }
}
```

```
Output:
False
True
```

Example 4: Program to match the given input is any special characters in a list.

```
using System;
using System.Text.RegularExpressions;
class Program
{
    static void Main(string[] args)
    {
        string pattern = "[$@#&^*!~%]";
        Regex regex = new Regex(pattern);

        Console.WriteLine(regex.IsMatch("abc*@"));
        Console.WriteLine(regex.IsMatch("abc"));

        Console.ReadKey();
   }
}
```

```
Output:
True
False
```

Example 5: Program to match the given input is starts at a specific character and ends with a specific character.

For this program, we need to use special symbols called metacharacters in the pattern as given below

^ - asserts position at start of the string

$ - asserts position at the end of the string

. - matches any character

For example, to match the given input is “apple” or not

```
using System;
using System.Text.RegularExpressions;
class Program
{
    static void Main(string[] args)
    {
        string pattern = "^a...e$";
        Regex regex = new Regex(pattern);

        Console.WriteLine(regex.IsMatch("apple"));
        Console.WriteLine(regex.IsMatch("orange"));

        Console.ReadKey();
    }
}
```

Output:

```
True

False
```

Metacharacters

To specify regular expressions, metacharacters are used. Metacharacters are characters that are interpreted in a special way by a regex engine.

Some of the basic metacharacters are:

Metacharacters	Purpose
[ ] (Square Bracket)	specifies a set of characters you wish to match. [abc] - any string that contains any of the a, b, or c. [^abc] - any string that not contains any of the a, b, or c. [a-zA-z0-9] – any string that contains English letters and digits

. (dot)	A period specifies any single character
	
	Example:
	`string pattern = "...";`
	
	It matches any three characters due to three dots.
	
	`Regex regex = new Regex(pattern);`
	`Console.WriteLine(regex.IsMatch("ap"));`
	`Console.WriteLine(regex.IsMatch("ora"));`
	
	False
	True

^ (Carat)	The caret symbol ^ specifies the string starts with a certain character. For example, ^m - any string starts with 'm'
$ - Dollar	The dollar symbol $ specifies the string ends with a certain character. For example, k$ - any string ends with 'k'
| - OR	The vertical bar | is used as or operator. **For example,** regex – a | b matches - string that has either a or b
() - Parenthesis	Parenthesis () is used to group sub-patterns. **For example,** regex - (a | b | c) xz matches - any string that has either a or b or c followed by xz

Special Sequences

Special sequences make commonly used patterns easier to write.

Metacharacters	Purpose
\A	Matches if the specified characters are at the start of a string. Example \Athe the sun - Match

	In the - No Match
\b	Matches if the specified characters are at the beginning or end of a word. Example: \bfoo football - Matching basketball - No Match If pattern is foo\b any word in a string that has foo at the end.
\B	Matches if the specified characters are not at the beginning or end of a word. For example, regex - \Bfoo matches - any word in a string that doesn't have foo at the beginning.
\d	Matches any decimal digit. Equivalent to [0-9] Example pattern = “\d” 12abc3 - Match Abc - No Match
\D	Matches any non-decimal digit. Equivalent to [^0-9] Example: regex : “\D” 12abc3 - No Match Abc - Match
\s	Matches where a string contains any whitespace character. Equivalent to [\t\n\r] **Example,** Regex - “\s” Hello World - Match HelloWorld - No Match
\S	Matches where a string contains any non-whitespace character. Equivalent to [^ \t\n\r].
\w	Matches any alphanumeric character (digits and alphabets). Equivalent to [a-zA-Z0-9_].

\W	Matches any non-alphanumeric character. Equivalent to [^a-zA-Z0-9_]

Quantifiers

Metacharacters	Purpose
{} - Braces	The braces symbol {} is used to specify the range of repetitions of the pattern left to it. **For example,** regex - a{2,3} string that has minimum 2 a's and maximum 3 a's left to it Others: a{3} - Exactly 3 'a' s a{3,} - 3 or more 'a'
+ (plus)	The plus symbol + matches one or more occurrences of the pattern left to it. Example: [a-z]+ Matches one or more occurrences of any English letters a to z For example, regex - ma+t matches - string that has one or more numbers of a in between m and t
* - Star	The star symbol * matches zero or more occurrences of the pattern left to it. Example: [a-z]* Matches zero or more occurrences of any English letters a to z For example, regex - ca*t matches - string that has any number[including zero] of a in between c and t
? - Question Mark	The question mark symbol ? matches zero or one occurrence of the pattern left to it. For example, regex - ma?n

	matches - string that has one or zero number of a in between m and n

Exercise: Write a program to check the given regno is URK20CS1002 or ULK20CS100 format or not.

```
using System;
using System.Text.RegularExpressions;
class Program
{
    static void Main(string[] args)
    {
        string pattern = "U(R|L)K20CS\\d{4}";
        Regex regex = new Regex(pattern);
        Console.Write("Enter your regno: ");
        String regno = Console.ReadLine();
        if (regex.IsMatch(regno))
            Console.WriteLine("Valid Regno");
        else
            Console.WriteLine("Invalid Regno");
        Console.ReadKey();
    }
}
```

Exercise: Write a program to check the given phone is

- 10 digit number
- Starts with 7,8,9

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

class Program
{
    static void Main(string[] args)
    {
        string pattern = "^[789]\\d{9}$";
        Regex regex = new Regex(pattern);
        Console.Write("Mobile Number: ");
        String regno = Console.ReadLine();
        if (regex.IsMatch(regno))
```

```
                Console.WriteLine("Valid");
            else
                Console.WriteLine("Invalid");
            Console.ReadKey();

        }
    }
```

Exercise: Write a program to check the given password is

At least 1 lowercase letter.

At least 1 uppercase letter.

At least 1 digit.

At least 1 special symbol.

Minimum length of 8 characters.

```
using System;
using System.Text.RegularExpressions;
class Program
{
    static void Main(string[] args)
    {
        string pattern = "(?=.*\\d)(?=.*[a-z])(?=.*[A-
                                                Z])(?=.*[@$*!^&~]).{8,}";
        Regex regex = new Regex(pattern);
        Console.Write("Enter your Password: ");
        String regno = Console.ReadLine();
        if (regex.IsMatch(regno))
            Console.WriteLine("Valid Password");
        else
            Console.WriteLine("Invalid Password");
        Console.ReadKey();
   }
}
```