A **regular expression** is a pattern that could be matched against an input text. The .Net framework provides a regular expression engine that allows such matching. A pattern consists of one or more character literals, operators, or constructs.

### Constructs for Defining Regular Expressions

There are various categories of characters, operators, and constructs that lets you to define regular expressions. Click the following links to find these constructs.

- [Character escapes](#)
- [Character classes](#)
- [Anchors](#)
- [Grouping constructs](#)
- [Quantifiers](#)
- [Backreference constructs](#)
- [Alternation constructs](#)
- [Substitutions](#)
- [Miscellaneous constructs](#)

### The Regex Class

The Regex class is used for representing a regular expression. It has the following commonly used methods:

<table>
<thead>
<tr>
<th>Sr.No</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>public bool IsMatch(string input)</code></td>
</tr>
<tr>
<td></td>
<td>Indicates whether the regular expression specified in the Regex constructor finds a match in a specified input string.</td>
</tr>
<tr>
<td>2</td>
<td><code>public bool IsMatch(string input, int startat)</code></td>
</tr>
<tr>
<td></td>
<td>Indicates whether the regular expression specified in the Regex constructor finds a match in the specified input string, beginning at the specified starting position in the string.</td>
</tr>
<tr>
<td>3</td>
<td><code>public static bool IsMatch(string input, string pattern)</code></td>
</tr>
<tr>
<td></td>
<td>Indicates whether the specified regular expression finds a match in the specified input string.</td>
</tr>
<tr>
<td>4</td>
<td><code>public MatchCollection Matches(string input)</code></td>
</tr>
<tr>
<td></td>
<td>Searches the specified input string for all occurrences of a regular expression.</td>
</tr>
<tr>
<td>5</td>
<td><code>public string Replace(string input, string replacement)</code></td>
</tr>
<tr>
<td></td>
<td>In a specified input string, replaces all strings that match a regular expression pattern</td>
</tr>
</tbody>
</table>
with a specified replacement string.

6  **public string[] Split(string input**

Splits an input string into an array of substrings at the positions defined by a regular expression pattern specified in the Regex constructor.

For the complete list of methods and properties, please read the Microsoft documentation on C#.

**Example 1**

The following example matches words that start with 'S':

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

namespace RegExApplication
{
    class Program
    {
        private static void showMatch(string text, string expr)
        {
            Console.WriteLine("The Expression: "+expr);
            MatchCollection mc = Regex.Matches(text, expr);
            foreach (Match m in mc)
            {
                Console.WriteLine(m);
            }
        }

        static void Main(string[] args)
        {
            string str = "A Thousand Splendid Suns";
            Console.WriteLine("Matching words that start with 'S': ");
            showMatch(str, @"\bS\S*");
            Console.ReadKey();
        }
    }
}
```

When the above code is compiled and executed, it produces the following result:

Matching words that start with 'S':
The Expression: \bS\S*
Splendid
Suns

**Example 2**

The following example matches words that start with 'm' and ends with 'e':

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

namespace RegExApplication
{
    class Program
    {
        private static void showMatch(string text, string expr)
        {
            Console.WriteLine("The Expression: "+expr);
            MatchCollection mc = Regex.Matches(text, expr);
            foreach (Match m in mc)
```
```csharp
{    Console.WriteLine(m);
}

static void Main(string[] args)
{
    string str = "make maze and manage to measure it";
    Console.WriteLine("Matching words start with 'm' and ends with 'e':");
    showMatch(str, @"\bm\S*e\b");
    Console.ReadKey();
}
```

When the above code is compiled and executed, it produces the following result:

Matching words start with 'm' and ends with 'e':
The Expression: \bm\S*e\b
make
maze
manage
measure

**Example 3**

This example replaces extra white space:

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

namespace RegExApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            string input = "Hello   World   ";
            string pattern = "\s+";
            string replacement = " ";
            Regex rgx = new Regex(pattern);
            string result = rgx.Replace(input, replacement);
            Console.WriteLine("Original String: {0}", input);
            Console.WriteLine("Replacement String: {0}", result);
            Console.ReadKey();
        }
    }
}
```

When the above code is compiled and executed, it produces the following result:

```
Original String: Hello World
Replacement String: Hello World
```