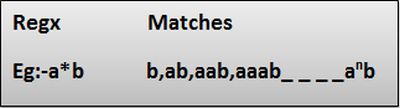
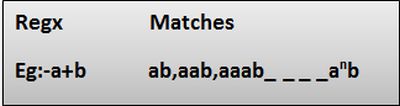
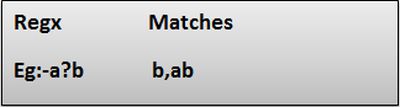
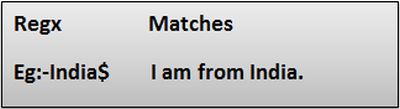
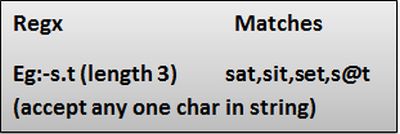
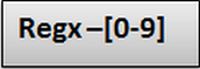
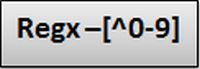
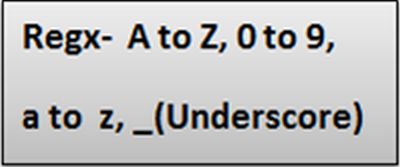
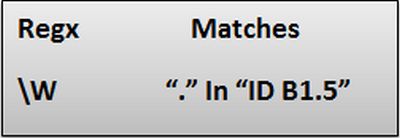
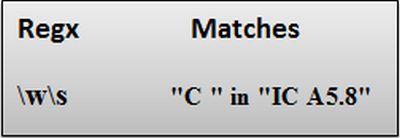
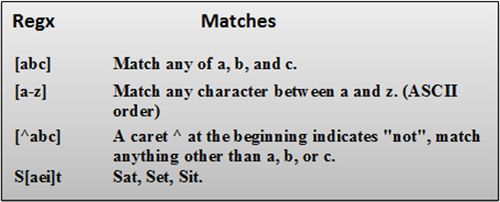
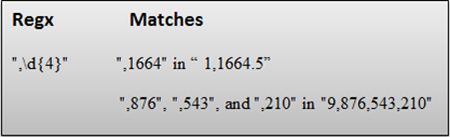
Regular Expression

Regular expressions are a pattern matching standard for string parsing and replacement and is a way for a computer user to express how a computer program should look for a specified pattern in text and then what the program is to do when each pattern match is found. Sometimes it is abbreviated "regex". They are a powerful way to find and replace strings that take a defined format.

Regex Syntax

The following basic syntax are used for regular expressions,

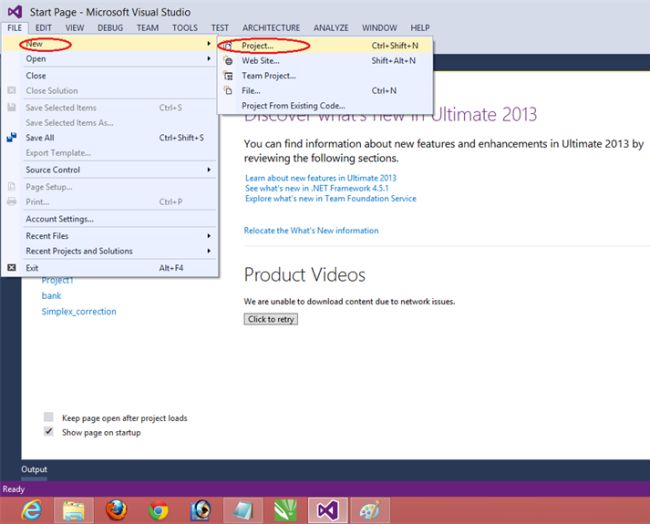
* **Quantifiers**The most important quantifiers are \*?+.  
    
  1. \* => Matches the preceding character zero or more times.  
       
     **Example**  
       
     
  2. + => Matches the preceding character 1 or more times.  
       
     **Example  
       
     **
  3. ? => Matches the preceding char zero or one time.  
       
     **Example  
       
     **
* **Special characters**Many special characters are available for regex building. Here are some of the more usual ones.  
    
  1. ^ => It is used to match the beginning of a string.  
      **Example**
  2. $=> It is used to match the end of a string.  
      **Example**
  3. (Dot) => Matches any character only once.  
      **Example**
  4. \d => It is used to match a digit character.  
       
     **Example**
  5. \D => It is used to match any non-digit character.  
       
     **Example**
  6. \w => It is used to match an alphanumeric character plus "\_".  
       
     **Example**
  7. \W => It is used to match any non-word character.  
       
     **Example**
  8. \s => Matches white space characters.  
       
     **Example**
  9. \S => Matches a non-white space character.  
       
     **Example**
  10. \n =>Matches a newline character.
* **Character classe**You can group characters by putting them between square brackets.This way, any character in the class will match one character in the input.  
    
  1. [ ] => It is used to match a range of characters.  
       
     **Example**
* **Grouping and alternatives**It's often necessary to group things together with parentheses ( and ).  
    
  1. ()=> It is used to group expressions.  
       
     **Example**  
     In the preceding image the | operator is the Or operator that takes any of the alternatives.
  2. {} =>It is used to match the preceding character for a specified number of times.  
       
     **i)** {n}=> Matches the previous element exactly n times.  
       
     **Example**  
       
       
       
     **ii)**{n,m} =>Matches the previous element at least n times, but no more than m times.  
       
     **Example**  
       
     **Example -**Regex for Mobile Number Validation

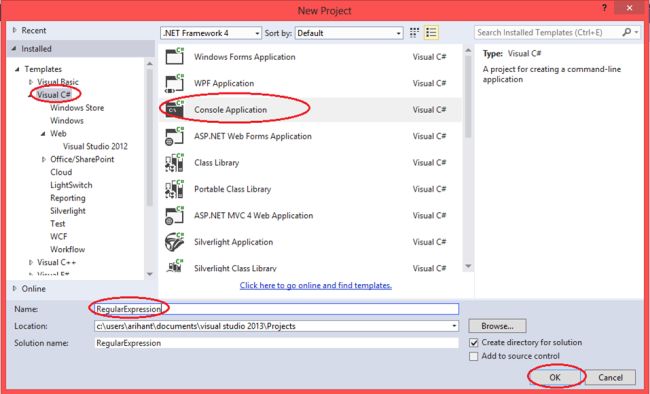
**Now** in the following procedure, I will explain how to create our own regular expression for Mobile Number validation in a C# console application.

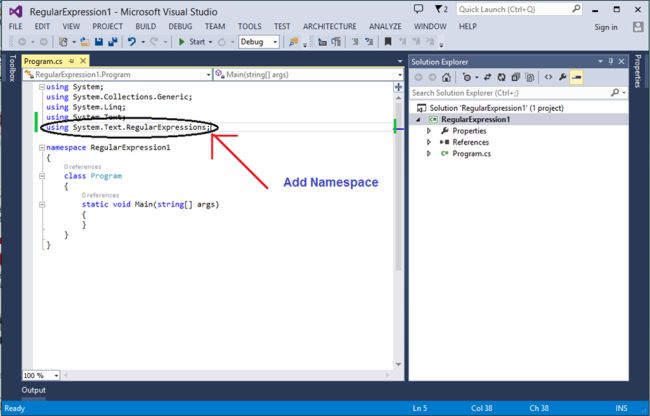
Step-by-step creation of a Regular Expression

**Step 1**

Open Visual Studio 2013.  
  
**Step 2**

Then click on "File" > "New" > "Project..." ( or press "Ctrl +Shift + N").  
  
  
  
**Step 3**

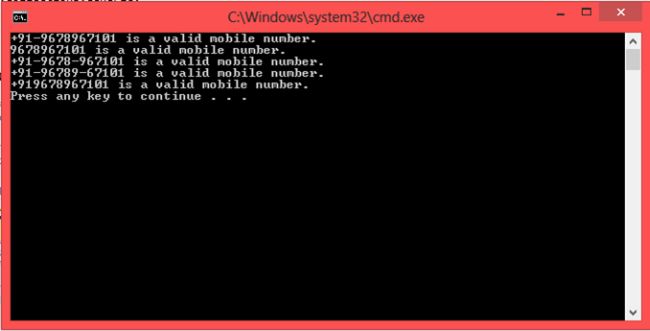
Then select Console Application, provide the name of the application "RegularExpression1" and click on OK.  
  
  
  
**Step 4**

After adding the project "RegularExpression1" you will see the following code in the "Program.cs" file and for creation of the regex add the namespace:  
  
**using System.Text.RegularExpressions;  
  
**  
  
**Step 5**

Now write the following code in the program.cs file to create the regex.

* 1. **using** System;
  2. **using** System.Collections.Generic;
  3. **using** System.Linq;
  4. **using** System.Text;
  5. **using** System.Text.RegularExpressions;
  7. **namespace** RegularExpression1
  8. {
  9. **class** Program
  10. {
  11. **static** **void** Main(**string**[] args)
  12. {
  13. Regex r = **new** Regex(@"^\+?\d{0,2}\-?\d{4,5}\-?\d{5,6}");
  14. //class Regex Repesents an immutable regular expression.
  16. **string**[] str = { "+91-9678967101", "9678967101", "+91-9678-967101", "+91-96789-67101", "+919678967101"};
  17. //Input strings for Match valid mobile number.
  19. **foreach**(**string** s **in** str)
  20. {
  21. Console.WriteLine("{0} {1} a valid mobile number.", s,
  22. r.IsMatch(s) ? "is":"is not");
  23. //The IsMatch method is used to validate a string or
  24. //to ensure that a string conforms to a particular pattern.
  25. }
  26. }
  27. }
  28. }

**Step 6**

After writing the code build program then run it and get the following output:  
  


Explanation of Regular Expression Pattern

It is one way for creation of a Mobile Number validation RegularExpression.  
  
Regular Expression Pattern 

