Ex. No. 4	String Manipulation and Regular Expression		
Date of Exercise	03.10.2016	Date of Upload	27.10.2016

Aim

To develop **Employee Information entry system** using C# by including the concept of various string functions and regular expressions.

Description

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.

The Regex Class

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

Sr.no	Methods
1	**public bool IsMatch(string input)** Indicates whether the regular expression specified in the Regex constructor finds a match in a specified input string.
2	**public bool IsMatch(string input, int startat)** 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.
3	**public static bool IsMatch(string input, string pattern)** Indicates whether the specified regular expression finds a match in the specified input string.
4	**public MatchCollection Matches(string input)**

	Searches the specified input string for all occurrences of a regular expression.
5	**public string Replace(string input, string replacement)** In a specified input string, replaces all strings that match a regular expression pattern 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.

It contains two features:

- A set of escape codes for identifying specific types of characters.
- A system for grouping parts of substrings and intermediate results during a search operation

Instantiate a System.Text.RegularExpressions.RegEx object, pass it the string to be processed, and pass in a regular expression.

With regular expressions, perform quite sophisticated and high - level operations on strings. For example,

- Identify all repeated words in a string
- Convert all words to title case
- Convert all words longer than three characters to title case
- Ensure that sentences are properly capitalized
- Separate the various elements of a URI

A regular expression string looks at first sight rather like a regular string, but interspersed with escape sequences and other characters that have a special meaning.

- the sequence \b indicates the beginning or end of a word
- to search for all occurrences of th at the end of a word, you would write th\b

Example

```
String text = “Here is a text!”;
Regex regExp = new Regex(@”\b[a-z]+\b”);
```

- + for one or more information
- * for o or more information

```
MatchCollection matches = regExp.Matches(text);
```

- Returns the matches in text with RE

```
foreach(Match m in matches)
{
if(m.Length!=0)
{ Console.WriteLine(m); }
}
```

The following table lists some of the main special characters or escape sequences that you can use. It is not comprehensive, but a fuller list is available in the MSDN documentation.

SYMBOL	MEANING	EXAMPLE	MATCHES
^	Beginning of input text	^B	B, but only if first character in text
$	End of input text	X$	X, but only if last character in text
.	Any single character except the newline character (\ )	i.ation	isation, ization
*	Preceding character may be repeated zero or more times	ra*t	rt, rat, raat, raaat, and so on
+	Preceding character may be repeated one or more times	ra+t	rat, raat, raaat and so on, but not rt
?	Preceding character may be repeated zero or one time	ra?t	rt and rat only
\s	Any whitespace character	\sa	[space]a, \ta, \na (\t and \n have the same meanings as in C#)
\S	Any character that isn't whitespace	\SF	aF, rF, cF, but not \tf
\b	Word boundary	ion\b	Any word ending in ion
\B	Any position that isn't a word boundary	\BX\B	Any X in the middle of a word

An example of this is **http://www.wrox.com:4355**

```
\b(\S+)://(\S+)(?::(\S+))?\b
```

- The first group, (\S+):// , identifies one or more characters that don ’ t count as whitespace, and that are followed by :// i.e http://
- The subsequent (\S+) identifies the string www.wrox.com in the URI
- The next group identifies the port (:4355)
- ? indicates that this group is optional in the match

Program

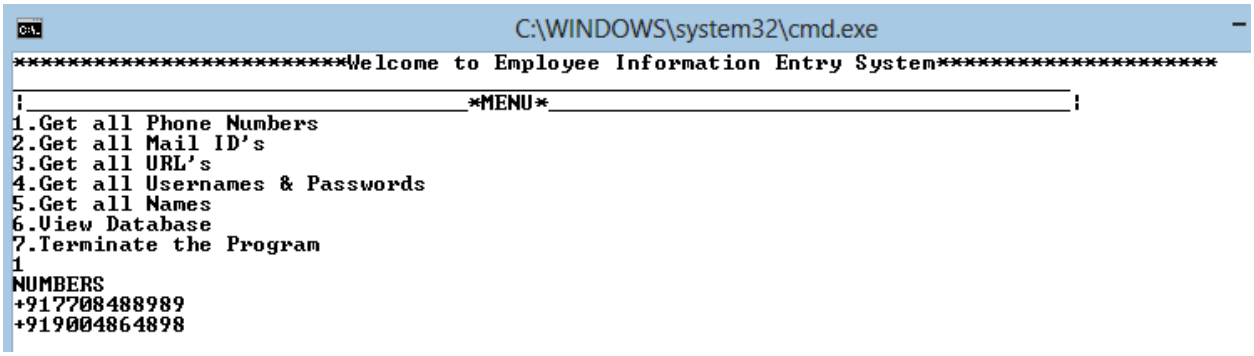
```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace Employee_Info_Entry
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("**********************Welcome to Employee Information
Entry System*********************");
            //  Readfromfile();
            Initialize_menu();
        }

        public static void Initialize_menu()
        {
            const string namepattern = @"^[A-Z][a-z]+$", numberpattern = @"\+\S+";
            const string mailpattern = @"(\S+)\@(\S+)\.(\S+)$", urlpattern =
@"\b(\S+)://([^:]+)(?:/(\S+))?(?::(\S+))?\b";
            const string userpattern = @"\$\S*\$", pwdpattern = @"\@\S*\@";
            while (true) {

Console.WriteLine("_____________________________________________________________________
_________");

Console.WriteLine("|_____________________________*MENU*_____________________________
_________|");
                Console.WriteLine("1.Get all Phone Numbers");
                Console.WriteLine("2.Get all Mail ID's");
                Console.WriteLine("3.Get all URL's");
                Console.WriteLine("4.Get all Usernames & Passwords");
                Console.WriteLine("5.Get all Names");
                Console.WriteLine("6.View Database");
                Console.WriteLine("7.Terminate the Program");
                int choice = Convert.ToInt32(Console.ReadLine());
                switch (choice)
                {
                    case 1:
                        Console.WriteLine("NUMBERS");
                        ReturnData(numberpattern);
                        break;
```

```
                    case 2:
                        Console.WriteLine("MAIL ID's");
                        ReturnData(mailpattern);
                        break;

                    case 3:
                        Console.WriteLine("URL'S");
                        ReturnData(urlpattern);
                        break;

                    case 4:
                        Console.WriteLine("Usernames\tPassword");
                        ReturnData(userpattern, pwdpattern);
                        break;

                    case 5:
                        Console.WriteLine("FNAME\tLASTNAME");
                        ReturnData(namepattern);
                        break;

                    case 6:
                        Console.WriteLine("FNAME\tLNAME\tMNO\t\tMail
Id's\t\tURL\t\t\tUsername    Pwd");
                        ReturnData(namepattern, numberpattern, mailpattern, urlpattern,
userpattern, pwdpattern);
                        break;

                    case 7:
                        Console.WriteLine("The program is terminated");
                        Environment.Exit(0);
                        break;
                    default:
                        Console.WriteLine("You have entered an Invalid Choice :( ");
                        Console.WriteLine("Please try again");
                        break;
                }
            }
        }

        public static void ReturnData(params string[] pattern) {
            try
            {
                // Create an instance of StreamReader to read from a file.
                // The using statement also closes the StreamReader.
                using (StreamReader sr = new
StreamReader("C:/Users/chinnu/Documents/Visual Studio
2015/Projects/CSharpLab/4.Strings_AND_RegEx[Employee info Entry
Sys]/Employee_Info_Entry/Employee.txt"))
                {
```

```
                    string line;

                    // Read and display lines from the file until
                    // the end of the file is reached.
                    while ((line = sr.ReadLine()) != null)
                    {
                        char tabdelem = '\t';
                        String[] splitfileds = line.Split(tabdelem);
                        foreach (string fieldval in splitfileds)
                        {
                            foreach (string currentpattern in pattern) {
                                if ((Regex.Match(fieldval, currentpattern)).Success)
                                {
                                    Console.Write(fieldval);
                                    if (pattern.Count() < 5)
                                    {
                                        Console.Write("\t");
                                    }
                                    else {
                                        Console.Write("  ");
                                    }
                                }

                            }
                        }
                        Console.Write("\n");
                     }
                }
            }
            catch (Exception e)
            {

                // Let the user know what went wrong.
                Console.WriteLine("The file could not be read:");
                Console.WriteLine(e.Message);
            }

        }

    }
}
```

Output

Phone Numbers



Mail Id's

URL's

Usernames

Username's and Password's

```
5
FNAME   LASTNAME
Michal  Bindhi
Guijue  Jeorge
```

First and Last Name's

```
6
FNAME   LNAME   MNO             Mail Id's              URL                        Username    Pwd
Michal  Bindhi  +917708488989   Michal99@gmail.com  https://git.com/broken-pot  $nuiasmoi$  @Mypwd@

Guijue  Jeorge  +919004864898   Guijue@gmail.com    https://www.karunya.edu:455  $snebidhi$  @merapwd@
```

Result

The above programmed is compiled successfully and the screenshots are well described with successful outputs and constraints.

[**Dr. J Anitha /Dr. S.P. Jeno Lovesum**]