Q.1 Attempt the following (any THREE) **[15]**

Q.1(a) Explain jagged array with an example. **[5]**

Ans.: A Jagged array is an array of arrays. You can declare a jagged array named scores of type int as int [][] scores;

Declaring an array, does not create the array in memory. To create the above array –

```
int[][] scores = new int[5][]; for (int i = 0; i < scores.Length; i++) {
  scores[i] = new int[4];
}
```

You can initialize a jagged array as -

```
int[][] scores = new int[2][]{new int[]{92,93,94},new int[]{85,66,87,88}};
```

Where, scores is an array of two arrays of integers - scores[0] is an array of 3 integers and scores[1] is an array of 4 integers.

Q.1(b) Explain various types of constructors in C#. **[5]**

Ans.: **What is a constructor?**

Constructor is a method which gets executed automatically when we create or instantiate object of that class having constructor.

Types of Constructors in C# :

There are 5 types of constructor in C# as listed below :

- Default Constructor
- Parameterized Constructor
- Copy Constructor
- Static Constructor
- Private Constructor

- **Default Constructor :** A constructor without any parameters is called as default constructor means a constructor which does not have any input parameter is known as default constructor.

```
public test()//Default Constructor
{   ……
}
```

- **Parameterized Constructor :** A constructor having one or more parameters is called as parameterized constructor means a constructor which is having single input parameter or multiple input parameters of same data types or different data types are known as parameterized constructor.

```
public test(double l, double b) // Parameterized constructor
{   ……
}
```

- **Copy Constructor :** A constructor that contains a parameter of same class type is called as copy constructor.

  C# does not provide a copy constructor. A copy constructor enables you to copy the data stored in the member variables of an object of the class into another new object means it helps to copy data stored in one object into another new object of the same instance.

```
class test
{
    public test(test t1) //Copy constructor
    {
```

```
            length = t1.length; breadth = t1.breadth;
        }
    }
```

- **Static Constructor :** Static constructor should be parameter less means it should not contain any input parameter.
  Program will not execute if static constructor is having any input parameter.
  Static constructor can be invoked once for any number instances are created and it is invoked only during the first initialization of instance. It is used to initialize static fields of the class Static constructor is created using a static keyword as shown below.

```
static int x;
static test()
{
    Console.WriteLine("static cons ");
    x = 10;//set the values for static member here
}
```

- **Private Constructor :** Private constructors are used to restrict the instantiation of object using 'new' operator.
  A private constructor is a special instance constructor. It is commonly used in classes that contain static members only. This type of constructors is mainly used for creating singleton object. If you don't want the class to be inherited we declare its constructor private.

```
private test() //private Constructor
{
    Console.WriteLine("private cons ");
}
```

Q.1(c) Explain Type Conversion in C#. **[5]**

Ans.:

- It is a process of converting value of one type into another type.
- It is also known as type casting.
- It is of two types :

1. Implicit type conversion:

- It is automatically done by C# in type safe manner.
- Conversion of lower type to higher type is done implicitly.
- No casting required.
- No loss of precision.

```
Example :  int x = 10;
           float y = x; // implicit
```

2. Explicit type casting:

- Here the cast operator/conversion function is explicitly required to perform casting.
- Conversion of higher type to lower type is done explicitly.
- There is loss of precision

```
Example:  double p=12.6;
          int x=(int)p;
```

- C# provides many conversion function. They are :
  1. ToBoolean – Converts a type to a Boolean where, possible.
  2. ToByte – Converts type to a byte.
  3. ToChar – Converts a type to a single unicode character, where possible.
  4. ToDatetime – Converts a type (integer to string) to date-time structure.
  5. ToDecimal – Converts floating point or integer type to decimal.

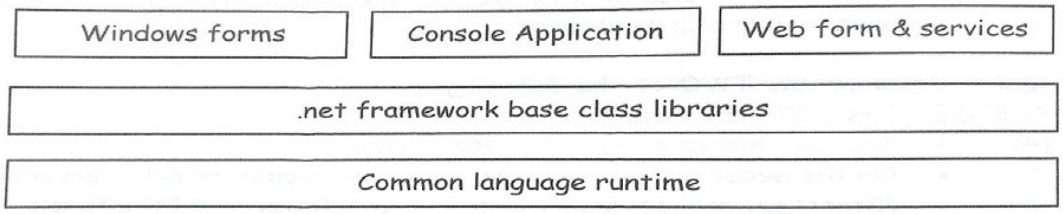
6. ToDouble – Converts a type to a double type.
7. ToInt16 – Converts a type to 16-bit integer.
8. ToInt32 – Converts a type to 32-bit integer.
9. ToInt64 – Converts a type to 64 bit integer.
10. ToSbyte – Converts a type to signed byte type.
11. ToSingle – Converts a type to small floating point number.
12. ToString – Converts a type of String
13. ToType – Converts a type of specified type.
14. ToUInt16 – Converts a type to unsigned int type.
15. ToUInt32 – Converts a type to unsigned long type.
16. ToUInt64 – Converts a type to an unsigned big int.

Note : These methods are static and called using class "Convert"

Q.1(d) Draw and Explain .NET framework architecture. **[5]**

Ans.: The .net framework consist of 3 main component

(i) CLR (Common Language Runtime)
(ii) .net framework base class
(iii) User and Program interface



(i) CLR:

- It is one of the most essential components of .net framework.
- It is the environment where all programs using .net technologies over executed.
- It allows the execution of code across different platform by transmitting code into (IL) Intermediate Language
- It is converted into machine language during execution by JIT(Just-In-Time) compiler.
- During JIT compilation the code is also check for type safety.
- It consist of common rules that must be followed by all language using .net framework these set of rules are called as common language specification(CLS).
- **One of the specification defined in CLS in common Type System (CTS).**
- CTS provides a type system that is common across all languages.

During execution CLR performs following steps:

- **Loading assemblies and identifies namespaces:**
  Assemblies are loaded into memory after loading assemblies the CLR will identifies namespaces for the code in the assemblies.
- **JIT Compilation**:
  Before execution the IL is converted into machine language using JIT compiler.
- **Garbage Collection:**
  Its process begins after the JIT compilation. It manages the allocation and deallocation of memory for an application.

(ii) .net framework class libraries:

- The .net framework class library works with any .net language such as vb.net, VC++.net and VC#. The libraryprovide classes that could be used in the code to perform:
  - String Management
  - data collection
  - file access

- database connectivity
- One of the most important features of .net class library is that it can be used in consistent manner across multiple languages.

(iii) User and program interface:

- At the presentation layer .net provide three types of user interface windows form, web forms and console application.

Q.1(e) Write short note on Value and Reference types. [5]

Ans.:

- Simple data types differ from more complex objects in one important way: simple data types are value types, while classes are reference types.
- This means that a variable for a simple data type contains the actual information you put in it (such as the number 7).
- On the other hand, object variables store a reference that points to a location in memory where the full object is stored.
- In most cases, .NET masks you from this underlying reality, and in many programming tasks you won't notice the difference.
- However, in three cases you will notice that object variables act a little differently than ordinary data types: in assignment operations, in comparison operations, and when passing parameters.

Boxing and Unboxing with reference to value type and reference type.

Boxing: Any type, value or reference can be assigned to an object without explicit conversion. When a compiler fined a value where it needs a reference type, it creates an object 'box' into which it places the value of the value type.

Example:

```
int m = 10;
object om = m;
```

When executed, this code creates a temporary reference _type 'box' for the object on heap. We can also use a C-style cast for boxing.

```
int m = 10;
object om = (object) m;
```

Note that the boxing operation creates a copy of the value of the m integer to the object om. Both the variables exist but the value of om resides on the heap. This means that the values are independent of each other.

Example

```
int m =10;
object om = m;
m = 20;
Console.WriteLine(m); // m= 20
Console .WriteLine(om); //om=10
```

Unboxing: Unboxing is the process of converting the object type back to the value type. Remember that a variable can be unboxed only if it has been previously boxed. In contrast to boxing, unboxing is an explicit operation.

Example:

```
int m = 100;
object om = m; //boxing
int n = (int) om; //unboxing
```

When performing unboxing, C# checks the value type we request is actually stored in the object under conversion. Only if it is, the value is unboxed.

Q.1(f) Explain static members and partial class.

Ans.: Static Members

- One of the tricks about .NET classes is that you really use them in two ways. You can use some class members without creating an object first. These are called *static* members, and they're accessed by class name. For example, the DateTime type provides a static property named Now. You can access this property at any time by using the full member name DateTime.Now. You don't need to create a DateTime object first.
- On the other hand, the majority of the DateTime members require a valid instance. For example, you can't use the AddDays() method or the Hour property without a valid object. These *instance* members have no meaning without a live object and some valid data to draw on.

The following code snippet uses static and instance members:

```
//Get the current date using a static property.
//Note that you need to use the class name DateTime.
DateTime myDate = DateTime.Now;

//Use an instance method to add a day.
//Note that you need to use the object name myDate.
myDate = myDate.AddDays(1);

//The following code makes no sense.
//It tries to use the instance method AddDays() with the class name DateTime! myDate =
DateTime.AddDays(1);
```

Both properties and methods can be designated as static. Static properties and methods are a major part of the .NET Framework, and you will use them frequently in this book. Some classes may consist entirely of static members, and some may use only instance members. Other classes, such as DateTime, provide a combination of the two.

Partial Classes

Partial classes give you the ability to split a single class into more than one C# source code (.cs) file. For example, if the Product class became particularly long and intricate, you might decide to break it into two pieces, as shown here:

This part is stored in file Product1.cs.

```
public partial class Product

{
public string Name { get; set; }
public event PriceChangedEventHandler PriceChanged; private decimal price;
public decimal Price
    {
Get
{
return price;
}
Set
{
price = value;
//Fire the event, provided there is at least one listener.
if (PriceChanged != null)
{
PriceChanged();
}
```

```
}
}
public string ImageUrl { get; set;}
public Product(string name, decimal price, string imageUrl)
{
Name = name;
Dr. Sarika Chouhan Page 8
Price = price;
ImageUrl = imageUrl;
}
}
//This part is stored in file Product2.cs.
public partial class Product

{
public string GetHtml()
{
string htmlString;
htmlString = "<h1>" + Name + "</h1><br />";
htmlString += "<h3>Costs: " + Price.ToString() +"</h3><br />"; htmlString += "<img src='" +
ImageUrl + "' />";
return htmlString;
}
}
```

A partial class behaves the same as a normal class. This means every method, property, and variable you've defined in the class is available everywhere, no matter which source file contains it. When you compile the application, the compiler tracks down each piece of the Product class and assembles it into a complete unit. It doesn't matter what you name the source code files, so long as you keep the class name consistent.

Partial classes don't offer much in the way of solving programming problems, but they can be useful if you have extremely large, unwieldy classes. The real purpose of partial classes in .NET is to hide automatically generated designer code by placing it in a separate file from your code. Visual Studio uses this technique when you create web pages for a web application and forms for a Windows application.

Q.2Attempt the following (any THREE) [15]

Q.2(a) List and explain any five templates to create ASP.NET Application. [5]

Ans.: You can create different types of ASP.NET applications by using various templates in Visual Studio.

ASP.NET Web Forms Site: This creates a full-featured ASP.NET website, with its basic infrastructure already in place. This website includes a master page that defines the overall layout (with a header, footer, and menu bar) and two ready-made web pages, named default.aspx and about.aspx. It also includes an Accounts folder with pages for registration, login, and password changing. After you've read this book and learned the details about all ASP.NET's web form features, you may choose to use this template as a starting point for a new site.

ASP.NET Web Site (Razor): This creates a website that uses the ASP.NET Model-View-Controller (MVC) approach, rather than the web forms model. The ASP.NET MVC model offers some benefits that ASP.NET web forms can't match—for example, it give developers more control over the way HTML is generated, and it creates websites that are far easier

to use with automated testing tools. The disadvantage is that ASP.NET MVC sites sacrifice some of traditional ASP.NET's convenience and ease of use.

ASP.NET Empty Web Site: This creates a nearly empty website. It includes a stripped-down web.config configuration file, and nothing else. Of course, it's easy to fill in the pieces you need as you start coding. This is the best starting point to learn about ASP.NET, because you won't have any extra, unnecessary, autogenerated files.

ASP.NET Dynamic Data Entities Web Site: This creates an ASP.NET website that uses the ASP.NET Dynamic Data feature. There are actually two dynamic data templates, which use slightly different approaches to communicating with your database.

ASP.NET 4.5 in C#.WCF Service: This creates a WCF service—a library of server-side methods that remote clients (for example, Windows applications) can call.

ASP.NET Reports Web Site: This creates an ASP.NET website that uses the ReportView control and SQL Server Reporting Services (a tool for generating database reports that can be viewed and managed over the Web). The ASP.NET Crystal Reports Web Site template provides a similar service, but it uses the competing Crystal Reports software.

Q.2(b) Explain The Page Life Cycle. [5]

Ans.: To understand how web control events work, we must understand the page life cycle. Following happens when a user changes a control that has the AutoPostBack property set to true :

1. On the client side, the JavaScript __doPostBack function is invoked, and the page is resubmitted to the server.
2. ASP.NET re-creates the Page object by using the .aspx file.
3. ASP.NET retrieves state information from the hidden view state field and updates the controls accordingly.
4. The Page.Load event is fired.
5. The appropriate change event is fired for the control. (If more than one control has been changed, the order of change events is undetermined.)
6. The Page.PreRender event fires, and the page is rendered (transformed from a set of objects to an HTML page).
7. Finally, the Page.Unload event is fired.
8. The new page is sent to the client.

Q.2(c) List and explain any four types of validation controls used in ASP.NET. [5]

Ans.: ASP.NET validation controls validate the user input data to ensure that useless, unauthenticated, or contradictory data don't get stored.

ASP.NET provides the following validation controls :

- RequiredFieldValidator
- CompareValidator
- CustomValidator
- RangeValidator
- RegularExpressionValidator
- ValidationSummary

- **RequiredFieldValidator Control :**

  The RequiredFieldValidator control is simple validation control, which checks to see if the data is entered for the input control. We can have a RequiredFieldValidator control for each form element on which you wish to enforce Mandatory Field rule.

  The syntax of the control is as given:

```
<asp:RequiredFieldValidator ID="rfvcandidate" runat="server"
 ErrorMessage="Please enter name !!" ControlToValidate="txtName">
</asp:RequiredFieldValidator>
```

- **RangeValidator Control :**
  The RangeValidator Server Control is another validator control, which checks to see if a control value is within a valid range. The attributes that are necessary to this control are: MaximumValue, MinimumValue, and Type.
  The syntax of the control is as given :
  <asp:RangeValidator ID="rvclass" runat="server"
  ControlToValidate="txtclass" 58
  ErrorMessage="Enter your class (6 - 12)" MaximumValue="12"
  MinimumValue="6" Type="Integer">
  </asp:RangeValidator>

- **RegularExpressionValidator :**
  The RegularExpressionValidator allows validating the input text by matching against a pattern of a regular expression. The regular expression is set in the ValidationExpression property.
  The syntax of the control is as given:
  <asp:RegularExpressionValidator ID="string" runat="server"
  ErrorMessage="string" ValidationExpression="string"
  ValidationGroup="string">
  </asp:RegularExpressionValidator>

- **CompareValidator Control :**
  The CompareValidator control allows you to make comparison to compare data entered in an input control with a constant value or a value in a different control.
  It can most commonly be used when you need to confirm password entered by the user at the registration time. The data is always case sensitive.
  It has the following specific properties:
  The basic syntax of the control is as follows:
  <asp:CompareValidator ID="CompareValidator1" runat="server"
  ErrorMessage="CompareValidator"> </asp:CompareValidator>

Q.2(d) Explain any five properties of List box and Drop-down list controls. [5]

Ans.:
- List boxes are used in cases where there are small numbers of items to be selected. In contrast, drop-down lists are typically used with larger list so that they don't take up much space on the page.
- A list box lets a user select one or more items from the list of items. A drop-down list lets a user choose an item from the drop-down list of items.

Following are common properties of ListBox and DropDownList:

Property	Description
Items	The collection of ListItem objects that represents the items in the control.
Rows	Specifies the number of items displayed in the box. If actual list contains more rows than displayed then a scroll bar is added.
SelectedIndex	The index of the currently selected item. If more than one item is selected, then the index of the first selected item. If no item is selected, the value of this property is -1.
SelectedValue	The value of the currently selected item. If more than one item is selected, then the value of the first selected item. If no item is selected, the value of this property is an empty string ("").
SelectionMode	Indicates whether a list box allows single selections or multiple selections.

Example:

The aspxcode for list box

```
<asp: ListBoxID="lstColor" runat="server">
<asp:ListItemvalue="Black"> Black</asp:ListItem>
<asp:ListItemvalue="red"> Red </asp:ListItem>
<asp:ListItemvalue="Blue"> Blue </asp:ListItem>
<asp:ListItemvalue="Green"> Green</asp:ListItem>
</asp:ListBox>
```

The aspxcode for Drop down list

```
<asp: DropDownListid="ddlDay" runat="server">
<asp:ListItemvalue="1"> Sunday</asp:ListItem>
<asp:ListItemvalue="2"> Monday</asp:ListItem>
<asp:ListItemvalue="3">Tuesday</asp:ListItem>
<asp:ListItemvalue ="4"> Wednesday </asp:ListItem>
<asp:ListItemvalue="5">Thursday</asp:ListItem>
<asp:ListItemvalue="6">Friday </asp:ListItem>
<asp:ListItemvalue="7">Saturday</asp:ListItem>
</asp:DropDownList>
```

Q.2(f) Explain AdRotator control with an example. [5]

Ans.:

- The basic purpose of the AdRotator is to provide a graphic on a page that is chosen randomly from a group of possible images.
- In other words, every time the page is requested, an image is selected at random and displayed, which is the rotation indicated by the name AdRotator.
- One use of the AdRotator is to show banner-style advertisements on a page, but you can use it anytime you want to vary an image randomly.

- The Advertisement File : The AdRotator stores its list of image files in an XML file. This file uses the format shown here:

```
<Advertisements>
<Ad>
<ImageUrl>prosetech.jpg</ImageUrl>
<NavigateUrl>http://www.prosetech.com</NavigateUrl>
<AlternateText>ProseTech Site</AlternateText>
<Impressions>1</Impressions>
<Keyword>Computer</Keyword>
</Ad>
</Advertisements>
```

- To add more advertisements, you would create multiple <Ad> elements and place them all inside the root <Advertisements> element.
- Each <Ad> element has a number of other important properties that configure the link, the image, and the frequency.

Element	Description
ImageUrl	The image that will be displayed. This can be a relative link (a file in the current directory) or a fully qualified Internet URL.
NavigateUrl	The link that will be followed if the user clicks the banner. This can be a relative or fully qualified URL.
AlternateText	The text that will be displayed instead of the picture if it cannot be displayed. This text will also be used as a tooltip in somenewer browsers.

Impressions	A number that sets how often an advertisement will appear. This number is relative to the numbers specified for other ads. For example, a banner with the value 10 will be shown twice as often (on average) as the banner with the value 5.
Keyword	A keyword that identifies a group of advertisements. You can use this for filtering. For example, you could create ten advertisements and give half of them the keyword Retail and the other half the keyword computer. The web page can then choose to filter the possible advertisements to include only one of these groups.

Q.2(e) Write a note on AdRotator control. **[5]**

Ans.:

- The basic purpose of the AdRotator is to provide a graphic on a page that is chosen randomly from a group of possible images.
- In other words, every time the page is requested, an image is selected at random and displayed,which is the *rotation* indicated by the name AdRotator.
- One use of the AdRotator is to show banner-style advertisements on a page, but you can use it anytime you want to vary an image randomly.
- **AdRotator Control Example in ASP.Net**

Step 1 – Open the Visual Studio –> Create a new empty Web application.

Step 2 – Create a New web page for display AdRotator control.

step 3 – Drag and drop AdRotator Control on web page from toolbox.

step 4 – go to **Add New Item** –> Add New **XML File** in project for write advertisement detail.

step 5 – Assign XML File to **AdvertisementFile** Property of AdRotator control.

The Advertisement File

The AdRotator stores its list of image files in an XML file. This file uses the format shown here:

```
<Advertisements>
<Ad>
<ImageUrl>prosetech.jpg</ImageUrl>
<NavigateUrl>http://www.prosetech.com
</NavigateUrl>
<AlternateText>ProseTech Site</AlternateText>
<Impressions>10</Impressions>
<Keyword>Computer</Keyword>
</Ad>
</Advertisements>
```

- To add more advertisements, you would create multiple <Ad> elements and place them all inside the root <Advertisements> element.
- Each <Ad> element has a number of other important properties that configure the link, the image, and the frequency.
- The actual AdRotator class provides a limited set of properties.
- You specify both the appropriate advertisement file in the AdvertisementFile property and the type of window that the link should follow (the Target window).
- The target can name a specific frame, or it can use one of the values.
- **This is a fully configured AdRotator tag:**
- <asp:AdRotator ID="Ads" runat="server" AdvertisementFile="MainAds.xml" Target="_blank" KeywordFilter="Computer" />

Q.3Attempt the following (any THREE) **[15]**

Q.3(a) What is user-defined exception? Explain with example. **[5]**

Ans.:

- We have seen built-in exception classes however, we often like to raise an exception when the business rule of our application gets violated.
- So, for this we can create a custom exception class by deriving Exception or ApplicationException class.
- Users can also define their own exception objects to represent custom error conditions.
- All you need to do is create an instance of the appropriate exception class and then use the throw statement.

Example : The given code will create user defined exception "bankexception" which will throw the exception when the balance amount is less than 500 into the account.

```
using System;
namespace ConsoleApplication1
{
    class BankException : Exception
    {
        public BankException(string msg)
        : base(msg)
        {
        }
    } class BankAccount
    {
        public int AccountNo;
        public double Balance;
        public BankAccount(int AcctNo, double Balance)
        {
            try
            {
                AccountNo = AcctNo;
                if (Balance < 500) throw new BankException("Oops!!!Lower Limit reached for
    Balance"); else
                this.Balance= Balance;
            }
            catch (BankException e)
            {
                Console.WriteLine(e.Message);
            }
            finally
            {
                Console.WriteLine("Account no:{0} and Balance {1}", AccountNo, Balance);
                Console.ReadKey();
            }
        }
    }
    class ExceptionHandling
    {
        static void Main(string[] args)
        {
            int a;
            double b;
            Console.Write("Enter Account Number:");
            a = int.Parse(Console.ReadLine());
            Account Balance:");
```

```
            b = double.Parse(Console.ReadLine());
            BankAccount obj = new BankAccount(a, b);
        }
    }
}
```

Output:

Enter Account Number:101
Enter Account Balance:350
Oops!!!Lower Limit reached for Balance
Account no:101 and Balance 350

Q.3(b) What is ViewState in ASP.NET? State its Advantages and Disadvantages. [5]

Ans.: View State is one of the most important and useful client side state management mechanisms. It can store the page value at the time of post back (Sending and Receiving information from Server) of your page. ASP.NET pages provide the ViewState property as a built-in structure for automatically storing values between multiple requests for the same page.

Example : If you want to add one variable in View State
Hide Copy Code ViewState["Var"]=Count;
For Retrieving information from View State
Hide Copy Code
string Test=ViewState["TestVal"];
Sometimes you may need to typecast ViewState Value to retreive. As I give an Example to strore and retreive object in view state in the last of this article.

Advantages : This are the main advantage of using View State:

- Easy to implement
- No server resources are required
- Enhanced security features, like it can be encoded and compressed.

Disadvantages : This are the main disadvantages of using View State:

- It can be performance overhead if we are going to store larger amount of data, because it is associated with page only.
- Its stored in a hidden filed in hashed format (which I have discussed later) still it can be easily trapped.
- It does not have any support on mobile devices.

Q.3(c) List and explain any 5 templates to create ASP.NET applications. [5]

Ans.: You can create different types of ASP.NET applications by using various templates in Visual Studio.

ASP.NET Web Forms Site: This creates a full-featured ASP.NET website, with its basic infrastructure already in place. This website includes a master page that defines the overall layout (with a header, footer, and menu bar) and two ready-made web pages, named default.aspx and about.aspx. It also includes an Accounts folder with pages for registration, login, and password changing. After you've read this book and learned the details about all ASP.NET's web form features, you may choose to use this template as a starting point for a new site.

ASP.NET Web Site (Razor): This creates a website that uses the ASP.NET Model-View-Controller (MVC) approach, rather than the web forms model. The ASP.NET MVC model offers some benefits that ASP.NET web forms can't match—for example, it give developers

more control over the way HTML is generated, and it creates websites that are far easier to use with automated testing tools. The disadvantage is that ASP.NET MVC sites sacrifice some of traditional ASP.NET's convenience and ease of use.

ASP.NET Empty Web Site: This creates a nearly empty website. It includes a stripped-down web.config configuration file, and nothing else. Of course, it's easy to fill in the pieces you need as you start coding. This is the best starting point to learn about ASP.NET, because you won't have any extra, unnecessary, autogenerated files.

ASP.NET Dynamic Data Entities Web Site: This creates an ASP.NET website that uses the ASP.NET Dynamic Data feature. There are actually two dynamic data templates, which use slightly different approaches to communicating with your database.

ASP.NET 4.5 in C#.WCF Service: This creates a WCF service—a library of server-side methods that remote clients (for example, Windows applications) can call.

ASP.NET Reports Web Site: This creates an ASP.NET website that uses the ReportView control and SQL Server Reporting Services (a tool for generating database reports that can be viewed and managed over the Web). The ASP.NET Crystal Reports Web Site template provides a similar service, but it uses the competing Crystal Reports software.

Q.3(d) Elaborate cookies with suitable code snippet. [5]

Ans.: A cookie is a name/value pair that's stored in the user's disk.

- A web application sends a cookie to a browser via an HTTP response.
- A browser sends a cookie to the server using HTTP Request.
- A *session cookie* is kept in the browser's memory and exists only for the duration of the browser's session.
- A persistent cookie is kept on the user's disk, retained until the cookie's expiration date.
- To create a cookie, you specify on its name or its name & value.
- To create a persistent cookie, you must also set the Expires property to the time when the cookie should expire.

Examples of cookies

- ASP.NET_SessionId = jsspvwpu553hcyx2w3jfa
- Email = anne_murach.com
- User_id = 4493
- Password = opensesame

Two ways to create a cookie

- new HttpCookie(name)
- new HttpCookie(name, value)

Code that creates a session cookie

```
HttpCookie nCookie = new HttpCookie("username", username);
```

Code that creates a persistent cookie

```
HttpCookie nCookie = new HttpCookie("username");
nCookie.Value = username;
nCookie.Expires = DateTime.Now.AddYears(1);
```

Cookie's common property:

- **Domain**: This is used to associate cookies to domain.
- **Secure**: We can enable secure cookie to set true (HTTPs).
- **Value**: We can manipulate individual cookie.

- **Values**: We can manipulate cookies with key/value pair.
- **Expires**: Which is used to set expire date for the cookies.

Advantages of Cookie:
- Its clear text so user can able to read it.
- We can store user preference information on the client machine.
- Its easy way to maintain.
- Fast accessing.

Disadvantages of Cookie
- If user clears cookie information we can't get it back.
- No security.
- Each request will have cookie information with page.

Q.3(e) Explain types of selectors in CSS. **[5]**

Ans.: Following are the types of Selectors in CSS :

1. **The Universal Selector :**
   The Universal selector, indicated by an asterisk (*), applies to all elements in your page. The Universal selector can be used to set global settings like a font family. The following rule set changes the font for all elements in your page to Arial :
   ```
   *{font-family: Arial;
   }
   ```

2. **The Type Selector :**
   The Type selector enables you to point to an HTML element of a specific type. With a Type selector, all HTML elements of that type will be styled accordingly.
   ```
   h1
   {
       color: Green;
   }
   ```
   The Type selector now applies to all <h1> elements in your code and gives them a green color type selectors are not case sensitive, so you can use booth h1 and H1 to refer to the same heading.

3. **The ID Selector :**
   The ID selector is always prefixed by a hash symbol (#) and enables you to refer to a single element in the page. Within an HTML or ASPX page, you can give an element a unique ID using the id attribute. With the ID selector, you can change the behavior for that single element, example :
   ```
   #IntroText
   {
       font-style: italic;
   }
   ```
   Because you can reuse this ID across multiple pages in your site (it only has to be unique within a single page), you can use this rule to quickly change the appearance of an element that you use once per page, but more than once in your site, for example with the following HTML code:
   ```
   <p id = "IntroText">I am italic because I have the right ID.</p>
   <p id = "BodyText"> I am NOT italic because I have a  different ID.</p>
   ```

   Here the #IntroText slector changes the font of the first paragraph which has the matching id attribute but leaves the other paragraph unmodified. ID selectors are case sensitive, ensure that id attribute and the selector always use the same casing.

4. **The Class Selector :**
   The Class selector enables you to style multiple HTML elements through the class attribute. This is handy when you want to give the same type of formatting to a number of unrelated HTML elements. The following rule changes the text to red and bold for all HTML elements that have their class attributes set to Highlight :

```
Highlight
{
    font-weight: bold; color: Red;
}
```

   The following code snippet uses the Highlight class to make the contents of a element and a link (<a>) appear with a bold typeface :

```
This is normal text but <span class = "Highlight"> this is Red and Bold.</span>
This is also normal text but <a href="CssDemo.aspx" class = "Highlight">this link is Red and Bold as well.</a>
```

5. ***Grouping and Combining Selectors :***
   CSS also enables you to group multiple selectors by separating them with a comma. This is handy if you want to apply the same styles to different elements. The following rule turns all headings in the page to red :

```
h1, h2, h3, h4, h5, h6
{
    color: Red;
}
```

   Moreover, with CSS you can also combine selectors, enabling you to hierarchically point to a specific element in a page. You can do this by separating the selectors with a space. The following example targets all <p> elements that fall within an element with an id of MainContent, leaving all other paragraphs unmodified.

```
#MainContent p
{
    font-size: 18px;
}
```

Q.3(f) What is cross page posting? Explain with an example. [5]

Ans.: *Cross-Page Posting*

- *A **cross-page postback*** is a technique that extends the postback mechanism so that one page can send the user to another page, complete with all the information for that page. This technique sounds conceptually straightforward, but it's a potential minefield. If you're not careful, it can lead you to create pages that are tightly coupled to others and difficult to enhance and debug.
- The infrastructure that supports cross-page postbacks is a property named PostBackUrl, which is defined by the IButtonControl interface and turns up in button controls such as ImageButton, LinkButton, and Button.
- To use cross-posting, you simply set PostBackUrl to the name of another web form. When the user clicks the button, the page will be posted to that new URL with the values from all the input controls on the current page.
- Here's an example—a page named CrossPage1.aspx that defines a form with two text boxes and a button. When the button is clicked, it posts to a page named CrossPage2.aspx.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="CrossPage1.aspx.cs"
Inherits="CrossPage1" %>
<html ="http://www.w3.org/1999/xhtml ">
    <head runat="server">
        <title>CrossPage1</title>
```

```
    </head>
    <body>
        <form id="form1" runat="server" >
            <div>
            First Name:
            <asp:TextBox ID="txtFirstName" runat="server"> </asp:TextBox>
            <br />
            Last Name:
            <asp:TextBox ID="txtLastName" runat="server"></asp:TextBox>
            <br />
            <br />
            <asp:Buttonrunat=" server" ID="cmdPost"
            PostBackUrl="CrossPage2.aspx" Text="Cross-Page Postback"/>
            <br />
        </div>
    </form>
</body>
</html>
```

The CrossPage 1 page doesn't include any code. Figure shows how it appears in the browser.

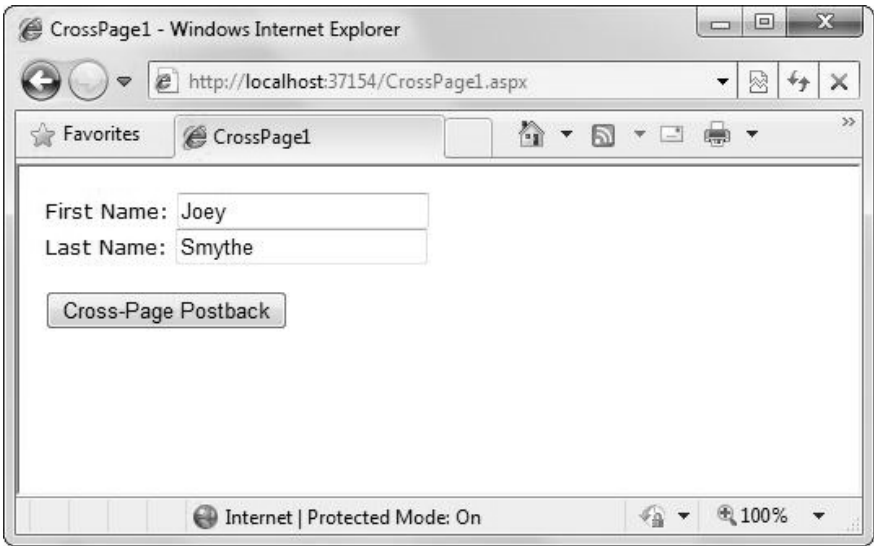


Fig. : *The starting point of a cross-page postback*

Now if you load this page and click the button, the page will be posted back to CrossPage2.aspx. At this point, the CrossPage2.aspx page can interact with CrossPage1.aspx by using the Page.PreviousPage property. Here's the code for the CrossPage2 page, which includes an event handler that grabs the title from the previous page and displays it:

```
public partial class CrossPage2 : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e)
    {
        if (PreviousPage != null)
        {
            lblInfo.Text = "You came from a page titled " +
            PreviousPage.Title;
        }
    }
}
```

Note that this page checks for a null reference before attempting to access the PreviousPage object. If it's a null reference , no cross-page postback took place. This means

CrossPage2.aspx was requested directly or CrossPage2.aspx posted back to itself. Either way, no PreviousPage object is available.

Figure shows what you'll see when CrossPage1.aspx posts to CrossPage2.aspx.

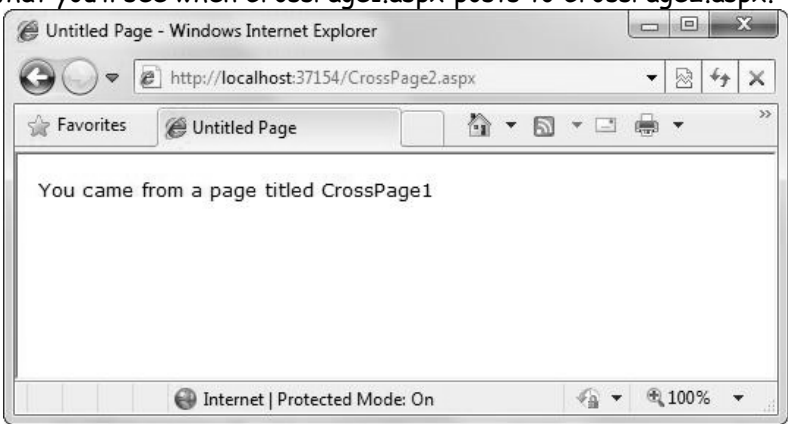


***Fig.:** The target of a cross-page postback*

Q.4Attempt the following (any THREE) **[15]**

Q.4(a) What is a GridView control? Explain with an example **[5]**

Ans.: The GridView control displays the values of a data source in a table. Each column represents a field, while each row represents a record. The GridView control supports the following features:

- Binding to data source controls, such as SqlDataSource.
- Built-in sort capabilities.
- Built-in update and delete capabilities.
- Built-in paging capabilities.
- Built-in row selection capabilities.
- Multiple key fields.
- Multiple data fields for the hyperlink columns.
- Customizable appearance through themes and styles.

Sorting allows the user to sort the items in the GridView control with respect to a specific column by clicking on the column's header. To enable sorting, set the AllowSorting property to true.

AllowSorting="True"

Instead of displaying all the records in the data source at the same time, the GridView control can automatically break the records up into pages. To enable paging, set the AllowPaging property to true.

AllowPaging="True"

Also we can set how many rows we want to see in a page.

PageSize-"4"

Example:

```
string constr = ConfigurationManager.ConnectionStrings["constr"].
ConnectionString;
using (SqlConnection con = new SqlConnection(constr))
{
using (SqlCommand cmd = new SqlCommand("SELECT * FROM Customers"))
{
cmd.Connection = con;
using (SqlDataAdapter sda = new SqlDataAdapter(cmd))
```

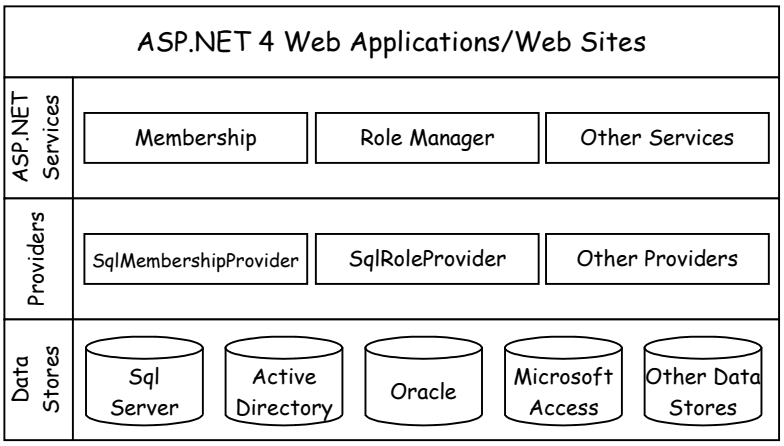
```
{
DataTable dt - new DataTable(); sda.Fill(dt);
GridView1.DataSource = dt;
GridView1.DataBind();
}
}
}
```

Q.4(b) What are the application services provided in ASP.NET? Explain. [5]

Ans.: ASP.NET 4 ships with a number of application services, of which the most important ones are :

- Membership : Enables us to manage and work with user accounts in our system.
- Roles : Enables us to manage the roles that your users can be assigned to.
- Profile : Enables us to store user-specific data in a back-end database.

Figure below gives an overview of these services and shows how they are related to our web site and the underlying data stores that the services may use.



A provider is software that provides a standardized interface between a service and a data source. ASP.NET providers are as follows:

- Membership
- Role Management
- Sitemap
- Profile
- Session state etc.

At the top of the diagram you see the ASP.NET 4 web sites and web applications that represent the web sites that you build. These web sites can contain controls like the login controls that in turn can talk to the ASP.NET application services such as membership and profile. To create a flexible solution, these services don't talk to an underlying data source directly, but instead talk to a configured provider.

A provider is an interchangeable piece of software that is designed for a specific task. For example, in the case of the membership services, the membership provider is designed to work with users in the underlying data store. You can configure different providers for the same application service depending on your needs.

Q.4(c) Explain SqlCommand class in detail. [5]

Ans.: **The SqlCommand class :** The process of interacting with a database means that we must specify the actions we want to occur. This is done with a command object. We use a command object to send SQL statements to the database.

A command object uses a connection object to figure out which database to communicate with. We can use a command object alone, to execute a command directly, or assign a reference to a command object to an SqlDataAdapter.

Properties of SqlCommand are :

Property	Description
Connection	The SqlConnection object used to connect to the database.
CommandText	The text of the SQL statement or the name of a stored procedure.
CommandType	A constant in the CommandType enumeration that indicates whether the CommandText property contains a SQL statement(Text) or the name of a stored procedure (StoredProcedure).
Parameters	The collection of parameters used by the command.

Methods of SqlCommand are :

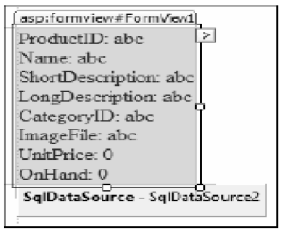
Method	Description
ExecuteReader	Executes a query and returns the result as a Sql DataReader object.
ExecuteNonQuery	Executes the command and returns an integer representing the number of rows affected.
ExecuteScalar	Executes a query and returns the first column of the first row returned by the query.

Q.4(d) Differentiate between FormView and DetailsView in ASP.NET. [5]

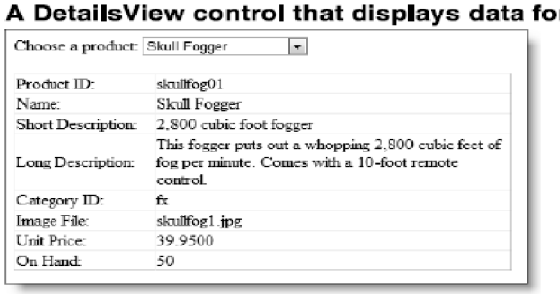
Ans.:

- Like DetailsView, FormView also displays a single record from the data source at a time.
- Both controls can be used to display, edit, insert and delete database records but one at a time. Both have paging feature and hence support backward and forward traversal.
- FormView is a new data-bound control that is nothing but a templated version of DetailsView control.
- The major difference between DetailsView and FormView is, here user need to define the rendering template for each item.
- The FormView control provides more formatting and layout options than DetailsView.
- The DetailsView control can uses <BoundField> elements or <TemplateField> elements to display bound data whereas FormView can use only templates to display bound data.
- The FormView control renders all fields in a single table row whereas the DetailsView control displays each field as a table row.
- When compare to DetailsView, the FormView control provides more control over the layout.

A FormView control after a data source has been assigned



A DetailsView control that displays data for a selected product



Q.4(e) Explain the SQL Data Provider Model. [5]

Ans.:

- ADO.NET relies on the functionality in a small set of core classes. You can divide these classes into two groups:
- Those that are used to contain and manage data (such as DataSet, DataTable, DataRow, and DataRelation) and those that are used to connect to a specific data source (such as Connection, Command, and DataReader).
- The data container classes are completely generic.
- No matter what data source you use, after you extract the data, it's stored using the same data container: the specialized DataSet class. Think of the DataSet as playing the same role as a collection or an array—it's a package for data.
- The difference is that the DataSet is customized for relational data, which means it understands concepts such as rows, columns, and table relationships natively.
- The second group of classes exists in several flavors. Each set of data interaction classes is called an ADO.NET data provider.
- Data providers are customized so that each one uses the best-performing way of interacting with its data source. For example, the SQL Server data provider is designed to work with SQL Server.
- Internally, it uses SQL Server's tabular data stream (TDS) protocol for communicating, thus guaranteeing the best possible performance.
- If you're using Oracle, you can use an Oracle data provider.
- Each provider designates its own prefix for naming classes. Thus, the SQL Server provider includes SqlConnection and SqlCommand classes, and the Oracle provider includes OracleConnection and OracleCommand classes.
- Internally, these classes work quite differently, because they need to connect to different databases by using different low-level protocols.
- Externally, however, these classes look quite similar and provide an identical set of basic methods because they implement the same common interfaces.
- This means your application is shielded from the complexity of different standards and can use the SQL Server provider in the same way the Oracle provider uses it.

Table : ADO.NET Namespaces for SQL Server Data Access

Namespace	Purpose
System.Data.SqlClient	Contains the classes you use to connect to a Microsoft SQL Server database and execute commands (such as SqlConnection and SqlCommand).
System.Data.SqlTypes	Contains structures for SQL Server-specific data types such as SqlMoney and SqlDateTime. You can use these types to work with SQL Server data types without needing to convert them into the standard .NET equivalents (such as System. Decimal and System.DateTime). These types aren't required, but they do allow you to avoid any potential rounding or conversion problems that could adversely affect data.
System.Data	Contains fundamental classes with the core ADO.NET functionality. These include DataSet and DataReladon, w[r]hich allow you to manipulate structured relational data. Ihese classes are totally independent of any specific type of database or the way you connect to it

Q.4(f) Give details about DataReader with example. [5]

Ans.:

- DataReader provides an easy way for the programmer to read data from a database as if it were coming from a stream.
- The DataReader is the solution for forward streaming data through ADO.NET.
- A data reader provides read-only, forward-only access to the data in a database.
- DataReader is also called a firehose cursor or forward read-only cursor because it

moves forward through the data.

- The DataReader not only allows us to move forward through each record of database, but it also enables us to parse the data from each column.
- The DataReader class represents a data reader in ADO.NET.

Example:
This example read all the records from customer table using DataReader class.

```
string SQL = "SELECT * FROM Customers";
SqlConnection conn = new SqlConnection(ConnectionString);
// create a command object
SqlCommand cmd = new SqlCommand(SQL, conn);
conn.Open();

// Call ExecuteReader to return a DataReader SqlDataReader reader =
cmd.ExecuteReader();
Console.WriteLine("customer ID, Contact Name, " + "Contact Title, Address ");
Console.W riteLine("============================");
while (reader.Read())
{
Console.Write(reader["CustomerID"].ToString() + ", ");
Console.Write(reader["ContactName"].ToString() + ", ");
Console.Write(reader["ContactTitle"].ToString() + ", ");
Console.WriteLine(reader["Address"].ToString() + ", ");
}
//Release resources
reader.Close();
conn.Close();
```

Q.5Attempt the following (any THREE) **[15]**

Q.5(a) What do you mean by Impersonation in ASP.NET? Explain. **[5]**

Ans.:

- ASP.NET impersonation is used to control the execution of the code in authenticated and authorized client.
- It is not a default process in the ASP.NET application.
- It is used for executing the local thread in an application.
- ASP.NET impersonation is used to control the execution of the code in authenticated and authorized client.
- It is not a default process in the ASP.NET application.
- It is used for executing the local thread in an application.
- If the code changes the thread, the new thread executes the process identity by default.

The impersonation can be enabled in two ways as mentioned below :

- Impersonation enabled: ASP.NET impersonates the token passed to it by IIS, can be an authenticated user or an internet user account.
- The syntax for enabling is as shown below:
  <identity impersonate="true" />
- Impersonation enabled for a specific identity: ASP.NET impersonates the token generated using the identity specified in the Web.config file.
  <identity impersonate="true" userName="domain\user"
  password="password"/>
- Impersonation disabled: It is the default setting for the ASP.NET application.
- The process identity of the application worker process is the ASP.NET account.
  <identity impersonate="false" />

Q.5(b) Create a web application to demonstrate the use of HTMLEditorExtender Ajax Control. Write the code of default.aspx and required property settings for the same. **[5]**

Ans.:

```
<%@Page Language="C#" AutoEventWireup="true"
CodeFile="default.aspx.cs" Inherits="_default" %>
<%@ Register Assembly="AjaxControlToolkit"
Namespace="AjaxControlToolkit.HtmlEditor.Sanitizer"
TagPrefix="ajaxToolkit" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<asp:TextBox runat="server" ID="txtBox1" TextMode="MultiLine"
Columns="50" Rows="10" Text="Hello <b>world!</b>" /><br />
<ajaxToolkit:HtmlEditorExtender ID="htmlEditorExtender1"
TargetControlID="txtBox1" runat="server"
DisplaySourceTab="true">
<Toolbar>
<ajaxToolkit:Undo />
<ajaxToolkit:Bold />
<ajaxToolkit:Italic />
<ajaxToolkit:Underline />
<ajaxToolkit:Copy />
<ajaxToolkit:JustifyLeft />
<ajaxToolkit:JustifyCenter />
<ajaxToolkit:JustifyRight />
<ajaxToolkit:Delete />
<ajaxToolkit:SelectAll />
<ajaxToolkit:FontColorSelector />
<ajaxToolkit:InsertImage />
</Toolbar>
</ajaxToolkit:HtmlEditorExtender>
</div>
</form>
</body>
</html>
```

Q.5(c) Explain the reading process from an XML Document with example. **[5]**

Ans.: Reading the XML document in your code is just as easy with the corresponding XmlTextReader class.

- The XmlTextReader moves through your document from top to bottom, one node at a time.
- You call the Read() method to move to the next node. This method returns true if there are morenodes to read or false once it has read the final node.
- The current node is provided through the properties of the XmlTextReader class, such as NodeType and Name.

- *A* node is a designation that includes comments, whitespace, opening tags, closing tags, content, and even *the XML declaration at the top of your file.*
- To get a quick understanding of nodes, you can use the XmlTextReader to run through your entire document from start to finish and display every node it encounters.

```
string file = Path.Combine(Request.PhysicalApplicationPath,
@"App_Data\SuperProProductList.xml");
FileStream fs = new FileStream(file, FileMode.Open);
XmlTextReader r = new XmlTextReader(fs);
// Use a StringWriter to build up a string of HTML that
// describes the information read from the XML document.

StringWriter writer = new StringWriter();
// Parse the file, and read each node.
while (r.Read())
{
// Skip whitespace.
if (r.NodeType == XmlNodeType.Whitespace) continue;
writer.Write("<b>Type:</b> ");
writer.Write(r.NodeType.ToString()); writer.Write("<br>");
// The name is available when reading the opening and closing tags // for an
element. It's not available when reading the inner content.
if (r.Name != "")
{
writer.Write("<b>Name:</b> ");
writer.Write(r.Name);
writer.Write("<br>");
}
// The value is when reading the inner content.
if (r.Value != "")
{
writer.Write("<b>Value:</b> ");
writer.Write(r.Value);
writer.Write("<br>");
}
if (r.AttributeCount > 0)
{
writer.Write("<b>Attributes:</b> "); for (int i = 0; i < r.AttributeCount; i++)
{
writer.Write(" "); writer.Write(r. GetAttribute(i)); writer.Write(" ");
}
writer.Write("<br>");
}
writer.Write("<br>");
}
fs.Close();
// Copy the string content into a label to display it. lblXml.Text = writer.ToString();
```

Q.5(d) What is Windows Authentication? Give details. **[5]**

Ans.: Windows-based authentication:

- It causes the browser to display a login dialog box when the user attempts to access restricted page.
- It is supported by most browsers.
- It is configured through the IIS management console.

- It uses windows user accounts and directory rights to grant access to restricted pages
  <authentication mode = "Windows">
  <forms name = "AuthenticationDemo" loginUrl = logon.asx protection = "All" path = "/" timeout = 30/>
  <authentication>

Steps to use Windows authentication:
When we configure ourASP.NET application as windows authentication it will use local window's user and groups to do authentication and authorization for our ASP.NET pages.
In 'web.config' file set the authentication mode to 'Windows' as shown in the below code snippets.

Authentication mode = "Windows"/>
We also need to ensure that all users are denied except authorized users. The below' code snippet inside the authorization tag that all users are denied. '?' indicates any
<authorization>
<deny users="?"/>
<authorization>

We also need to specify the authorization part. We need to insert the below' snippet in the 'web.config' file .stating that only 'Administrator' users will have access to
<location path= "Admin. aspx">
<system.web>
<authorization>
<allow roles="questpon-srize2\Administrator"/>
<deny users="*"/>
<authorization>
</system.web>
</location>

The next step is to compile the project and upload the same on an IIS virtual directory. On the IIS virtual directory we need to ensure to remove anonymous access and check the integrated windows authentication.
Now' if we run the web application we will be popped with a userid and password box.

Q.5(e) Explain the UpdatePanel Control in ASP.Net. [5]

Ans.: The update Panel control is most important control for creating flicker-free pages we just wrap the control around the content that we want to update and add the script manager to the page.

Properties of UpdatePanel Control :

Property	Description
ChildrenAsTriggers	This property determines whether controls located within the UpdatePanel can cause a refresh of the UpdatePanel. The default value is True. When you set this value to False, you have to set the UpdateMode to Conditional.
Triggers	The Triggers collection contains PostBackTrigger and AsyncPostBackTrigger elements.
RenderMode	This property can be set to Block or Inline to indicate whether the UpdatePanel renders itself as a or element
UpdateMode	This property determines whether the control is always refreshed (the UpdateMode is set to Always) or only under certain conditions, for example,when one of the controls defined in the <Triggers> element is causing a postback (the UpdateMode is set to Conditional).
ContentTemplate	The <ContentTemplate> is an important property of the UpdatePanel. It"s the container in which you place controls as children of the UpdatePanel.

General Form :

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <div>
        </div>
    </ContentTemplate>
</asp:UpdatePanel>
```

Q.5(f) Explain AJAX with its advantages and Disadvantages. [5]

Ans.: AJAX (Asynchronous JavaScript and XML) is not a single technology, it is a collection of technologies that, when used together, enable developers to build an RIA.

- These technologies allow a web page to communicate with a web server and update the page after it's loaded without having to reload the entire page. This can also reduce the load on the web server.
- Each time a standard HTTP request and response cycle is performed, the entire page is returned from the server and the page is loaded into the browser.
- This type of request and response is required the first time a page is requested even if the page is AJAX-enabled.
- With an AJAX HTTP request and response cycle, the browser can request just the information it need to update the page.
- Then, the updated information that's returned from the server can be used to update the page without having to reload it.

Advantages:

- The key benefit of Ajax is responsiveness. An Ajax application, when done properly, provides a better experience for the user.
- Even if the user can't do anything new (or do anything faster), this improved experience can make your web application seem more modern and sophisticated.
- Ajax can also provide genuinely new features that aren't possible in traditional web pages.
- For example,Ajax pages often use JavaScript code that reacts to client-side events such as mouse movements and key presses.
- These events occur frequently, so it's not practical to deal with them by using the postback model.
- With the postback approach,you'd need to send the entire page back to the web server, regenerate it, and refresh it in the browser—by which point the mouse might be somewhere completely different. This approach is clearly impractical.
- However, an Ajax page can deal with this scenario because it can react immediately, updating the page if needed or requesting additional information from the web server in the background. While this request is under way, the user is free to keep working with the page. In fact, the user won't even realize that the request is taking place.

Disadvantages:

- There are two key challenges to using Ajax.
- The first is complexity. Writing the JavaScript code needed to implement an Ajax application is a major feat.
- The other challenge to using Ajax is browser support. In the past, this was a significant concern, but today its, rare to find a browser that doesn't support the necessary JavaScript features, even on mobile devices.
- ASP.NET is clever enough to use fallbacks when it encounters a browser that doesn't support Ajax (or has JavaScript switched off).
- Finally, Ajax applications introduce a few quirks that might not be to your liking. Web pages that use Ajax often do a lot of work on a single page. This is different from traditional web pages, which often move the user from one page to another to complete

a task.

- Although the multiple-page approach is a little more round about,it allows the user to place bookmarks along the way and use the browser's Back and Forward buttons to step through the sequence.
- These techniques usually don't work with Ajax applications, because there's only a single page to bookmark or navigate to, and the URL for that page doesn't capture the user's current state.

❑ ❑ ❑ ❑ ❑