1. Attempt *any two* of the following. 10

a) Explain the similarities and differences between Interfaces and Abstract classes(Any 5 similarities and differences in paragraph or distinguish form 5 marks)

Ans) Similarities: Both abstract classes and interfaces may contain members that can be inherited by a derived class. Neither interfaces nor abstract classes may be directly instantiated, but you can declare variables of these types. If you do, you can use polymorphism to assign objects that inherit from these types to variables of these types. In both cases, you can then use the members of these types through these variables, although you don't have direct access to the other members of the derived object.

Differences: Derived classes may only inherit from a single base class, which means that only a single abstract class can be inherited directly (although it is possible for a chain of inheritance to include multiple abstract classes). Conversely, classes can use as many interfaces as they want
Abstract classes may possess both *abstract members* and *non-abstract members* (these possess a code body, and can be virtual so that they may be overridden in the derived class). *Interface members* conversely, must be implemented on the class that uses the interface — they do not possess code bodies. Moreover, interface members are by definition public but members of abstract classes may also be private (as long as they aren't abstract), protected, internal, or protected internal (where protected internal members are accessible only from code within the application or from a derived class). In addition, interfaces can't contain fields, constructors, destructors, static members, or constants.

b) Write a short note on Assembly.
(Correct answer 5 marks)

Ans) When you compile an application, the CIL code created is stored in an *assembly*. Assemblies include both executable application files that you can run directly from Windows without the need for any other programs (these have a .exe file extension) and libraries (which have a .dll extension) for use by other applications.
In addition to containing CIL, assemblies also include *meta* information (that is, information about the information contained in the assembly, also known as *metadata*) and optional *resources* (additional data used by the CIL, such as sound files and pictures). The meta information enables assemblies to be fully self-descriptive. You need no other information to use an assembly, meaning you avoid situations such as failing to add required data to the system registry and so on, which was often a problem when developing with other platforms. This means that deploying applications is often as simple as copying the files into a directory on a remote computer. Because no additional information is required on the target systems, you can just run an executable file from this directory and (assuming the .NET CLR is installed) you're good to go. Of course, you won't necessarily want to include everything required to run an application in one place. You might write some code that performs tasks required by multiple applications. In situations like that, it is often useful to place the reusable code in a place accessible to all applications. In the .NET Framework, this is the *global assembly cache (GAC)*. Placing code in the GAC is simple — you just place the assembly containing the code in the directory containing this cache.

c) Give syntax of foreach loop. Explain with example.
(syntax with explanation 2.5 marks , example 2.5 marks)

Ans) A foreach loop enables you to address each element in an array using this simple syntax:

```
foreach (<baseType> <name> in <array>)
{
// can use <name> for each element
}
```

This loop will cycle through each element, placing it in the variable <*name*> in turn, without danger of accessing illegal elements. You don't have to worry about how many elements are in the array, and you can be sure that you'll get to use each one in the loop. The main difference between using this method and a standard for loop is that foreach gives you *read-only* access to the array contents, so you can't change the values of any of the elements.

Example

```
static void Main(string[] args)
{
string[] friendNames = { "Robert Barwell", "Mike Parry", "Jeremy Beacock" };
Console.WriteLine("Here are {0} of my friends:", friendNames.Length);
foreach (string friendName in friendNames)
{
Console.WriteLine(friendName);
}
Console.ReadKey();
}
```

d) Write a program using overloaded constructors.
(Correct example 5 marks)

Ans)

```
using System;
class Room
{
   public double length;
   public double breadth;

public Room(double x, double y)
{
   length=x;
   breadth=y;
 }
public Room( double x)
{
     length=breadth=x;
}
public void  Area()
{
  double a=length*breadth;
 Console.WriteLine("Area is" + a);
}

class Mainclass
{
public static void Main(string[] args )
{
   Room room1=new Room(25.0,15.0);
   Room room2= new Room(20.0);
     room1.Area();
     room2.Area();
 Console.ReadKey();
}
}
```

2. Attempt *any two* of the following: 10

a) Explain Boxing and Unboxing with reference to value type and reference type.
(Boxing 2.5 marks, Unboxing 2.5 marks

Ans) Boxing : Any type, value or reference can be assigned to an object without explicit conversion. When a compiler find a value where it needs a reference type, it creates an object ‘box’ into which it places the value of the value type. For 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 otheAnsr. For 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.

```
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.

b) Explain the four most important selectors present in CSS.

Ans)

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;
}
```

This 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 both 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, for 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>
```

In this example, the #IntroText selector changes the font of the first paragraph — which has the matching id attribute — but leaves the other paragraph unmodified. ID selectors are case sensitive, so make sure that the 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 <span> 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>
```

Notice that the selector uses a period in its name, but you don’t use this period when referring to the selector in the class attribute. The class attribute is very useful because it enables you to reuse a piece of CSS for many different purposes, regardless of the HTML element that uses the class.

c) Explain the different types of CSS present in ASP.NET.

Ans)

1) External style sheet :- The first way to add CSS style sheets to your web pages is through the <link> element that points to an *external* CSS file. For example the following <link> shows what options you have when embedding a style sheet in your page:

```
<link href=”StyleSheet.css” rel=”Stylesheet” type=”text/css” media=”screen” />
```

The href property points to a file within your site, just as you saw in the previous chapter when you created links between two pages. The rel and type attributes tell the browser that the linked file is in fact a cascading style sheet. The media attribute is quite interesting: it enables you to target different devices, including the screen, printer, handheld devices, and even Braille and aural support tools for visually impaired visitors. The default for the media attribute is screen, so it’s OK to omit the attribute if you’re targeting standard desktop browsers.

2) Embedded style sheet:-The second way to include style sheets is using *embedded* <style> elements. The <style> element should be placed at the top of your ASPX or HTML page,between the <head> tags. For example, to change the appearance of an <h1> element in the current page alone, you can add the following code to the <head> of your page:

```
<head runat=”server”>
<title></title>
<style type=”text/css”>
h1
{
color: Blue;
}
</style>
</head>
```

3) Inline style sheet :- The third way to apply CSS to your HTML elements is to use *inline styles*.. Because the style attribute is already applied to a specific HTML element, you don't need a selector and you can write the declaration in the attribute directly:

```
<span style="color: White; background-color: Black;">
This is white text on a black background.
</span>
```

d) Write a program using any five Methods/Property of ArrayList Class.
(Each propery or method used correctly 1 mark each)

Ans)

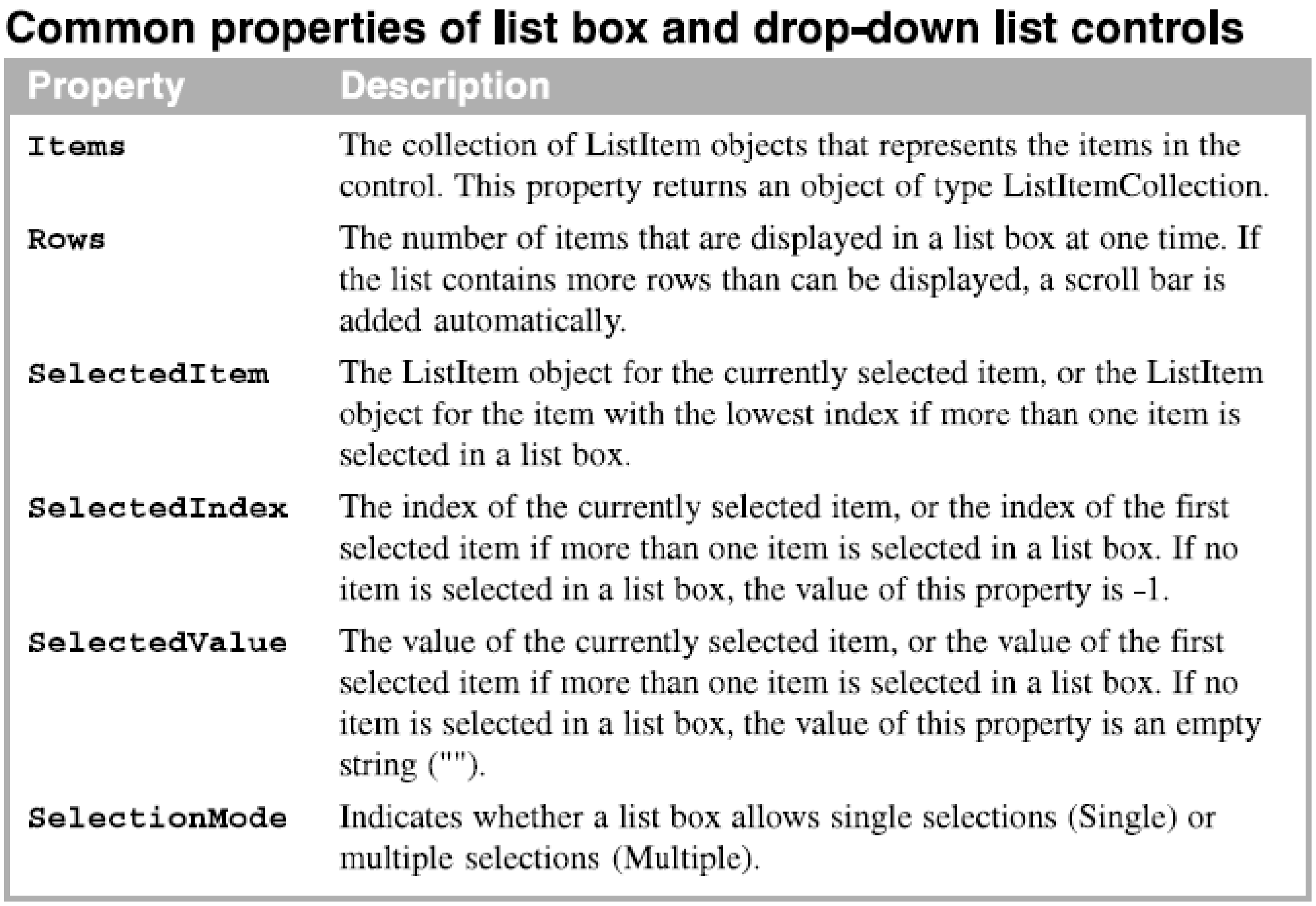
```
using System;
using System.Collections;
using System.Text;
class Program
{
  public static void Main( string[] args)
   {
     Arraylist n = new ArrayList();
    n.Add("Mumbai");
    n.Add("Pune");
    n.Add("Kolkatta");
    n.Add("Delhi");
    n.Add("Chennai');
     Console.Writeline( "ArrayList has capacity : " +n.Capacity);
    Console.WriteLine( " ArrayList has count : " + n.Count);
     Console.WriteLine();
          n.Sort();
      for( int i=0; i<n.Count;i++)
        {
          Console.Writeline(n[i]);
         }
     n.RemoveAt(3);
       for( int i=0; i<n.Count;i++)
        {
          Console.Writeline(n[i]);
         }
     Console.ReadKey();
   }
 }
```

3. Attempt *any two* of the following: 10

a) What is the difference between List Box and Drop-Down Lists? List and explain any three common properties of these controls.(Difference 2 marks, any 3 properties 3 marks)

Ans) i) List boxes are used in cases where there are small number 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.

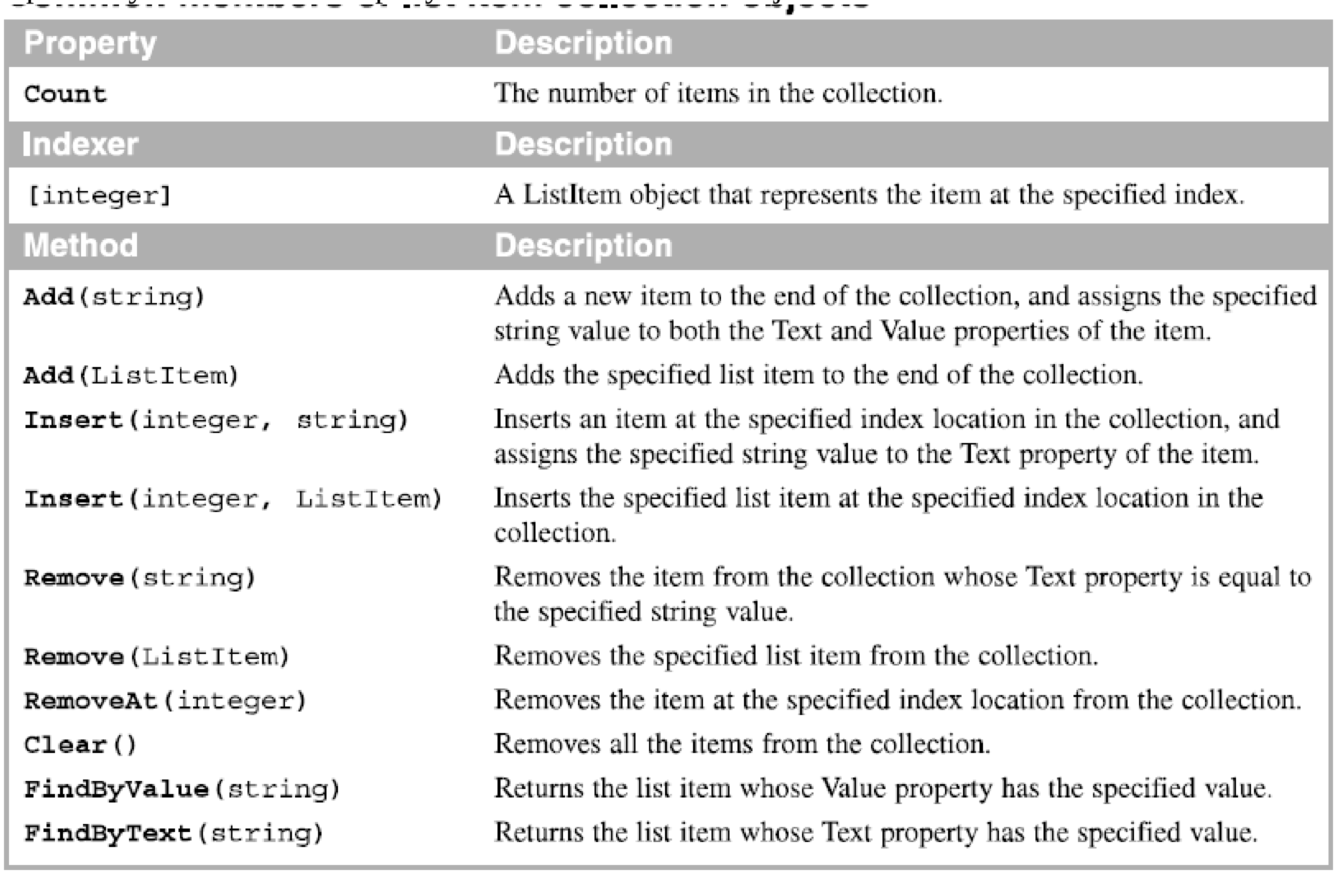
ii) A list box lets a user select one or more items from the list of items.A drop-down list lets a user choose an item from the drop-down list of items.



Common properties of list box and drop-down list controls

Property	Description
`Items`	The collection of ListItem objects that represents the items in the control. This property returns an object of type ListItemCollection.
`Rows`	The number of items that are displayed in a list box at one time. If the list contains more rows than can be displayed, a scroll bar is added automatically.
`SelectedItem`	The ListItem object for the currently selected item, or the ListItem object for the item with the lowest index if more than one item is selected in a list box.
`SelectedIndex`	The index of the currently selected item, or the index of the first selected item if more than one item is selected in a list box. If no item is selected in a list box, the value of this property is -1.
`SelectedValue`	The value of the currently selected item, or the value of the first selected item if more than one item is selected in a list box. If no item is selected in a list box, the value of this property is an empty string ("").
`SelectionMode`	Indicates whether a list box allows single selections (Single) or multiple selections (Multiple).

b) Explain any five Methods/Property of List Item collection objects.



Property	Description
`Count`	The number of items in the collection.
Indexer	**Description**
`[integer]`	A ListItem object that represents the item at the specified index.
Method	**Description**
`Add(string)`	Adds a new item to the end of the collection, and assigns the specified string value to both the Text and Value properties of the item.
`Add(ListItem)`	Adds the specified list item to the end of the collection.
`Insert(integer, string)`	Inserts an item at the specified index location in the collection, and assigns the specified string value to the Text property of the item.
`Insert(integer, ListItem)`	Inserts the specified list item at the specified index location in the collection.
`Remove(string)`	Removes the item from the collection whose Text property is equal to the specified string value.
`Remove(ListItem)`	Removes the specified list item from the collection.
`RemoveAt(integer)`	Removes the item at the specified index location from the collection.
`Clear()`	Removes all the items from the collection.
`FindByValue(string)`	Returns the list item whose Value property has the specified value.
`FindByText(string)`	Returns the list item whose Text property has the specified value.

c) List and explain the comparison and logical operators in C#.(2.5 comparison operators, 2.5 logical operators)

Ans) Comparison operators

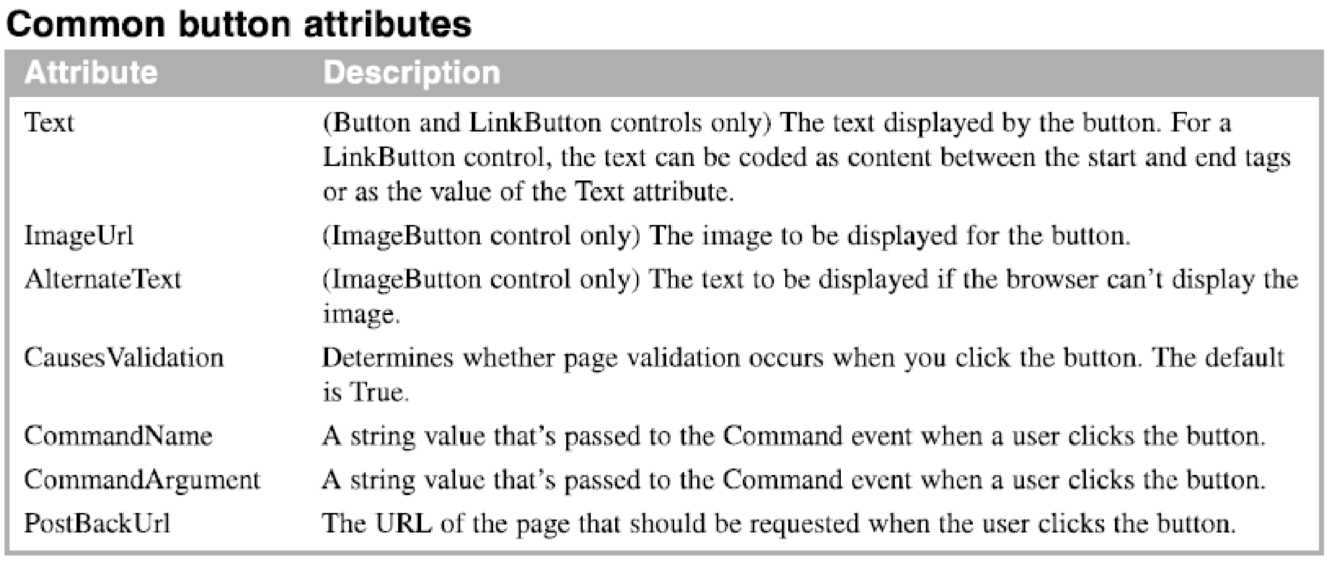
==	Checks if two values are equal to each other.
!=	Checks if two values are not equal
<	Checks if the first value is less than the second.
>	Checks if the first value is greater than the second.
<=	Checks if the first value is less than or equal to the second.
>=	Checks if the first value is greater than or equal to the second.

Logical operators

&	Returns True when both expressions result in a True value
|	Returns True if at least one expression results in a True value.
!	Reverses the outcome of an expression.
&&	Enables you to short-circuit your logical And condition checks.
||	Enables you to short-circuit your logical Or condition checks.

d) What is the difference between buttons, Link Buttons and Image Buttons? Explain any three common button attributes.

Ans) These controls differ only in how they appear to the user. A button displays text within a rectangular area. A Link button displays text that look like a hyperlink and an image button displays an image.



Common button attributes

Attribute	Description
Text	(Button and LinkButton controls only) The text displayed by the button. For a LinkButton control, the text can be coded as content between the start and end tags or as the value of the Text attribute.
ImageUrl	(ImageButton control only) The image to be displayed for the button.
AlternateText	(ImageButton control only) The text to be displayed if the browser can't display the image.
CausesValidation	Determines whether page validation occurs when you click the button. The default is True.
CommandName	A string value that's passed to the Command event when a user clicks the button.
CommandArgument	A string value that's passed to the Command event when a user clicks the button.
PostBackUrl	The URL of the page that should be requested when the user clicks the button.

4. Attempt *__any two__* of the following: **10**

a) Write a program to create a new cookie with the name "Username" and add it to the HttpResponse object on the click of a button. Set the expiry date of the cookie to One year from Now.

Ans)

```
 protected void Button1_Click(object sender, EventArgs e)
{
      HttpCookie Username = new HttpCookie("UserName", "WELCOME");
     Username.Expires=DateTime.Now.AddYears(1);
     Response.Cookies.Add(Username);
 }
```

b) What is the use of MasterPages in ASP.NET? How a Content page can be added to a Master Page.

Ans) A Master page provides a framework in which the contents of each page on a website is presented. Master pages make it easy to create pages that have a consistent look. Master pages make it easy to include banners, navigation menus, header, footer and other elements on all the pages in an

application. The pages that provide the content that's displayed in the master page are called content pages. The content of each content page is displayed in the master page's content placeholder.
There are two ways of adding content pages to master page.

i) Choose the website→ Add New Item Command. Then, select the web Form template. Enter the name for the form, check the Select Master Page check box and then click Add. When select a Master Page dialog box appears, select the master page you want and click OK.

ii) Select the Master Page in the Solution Explorer, then choose the Website → Add Content page Command.

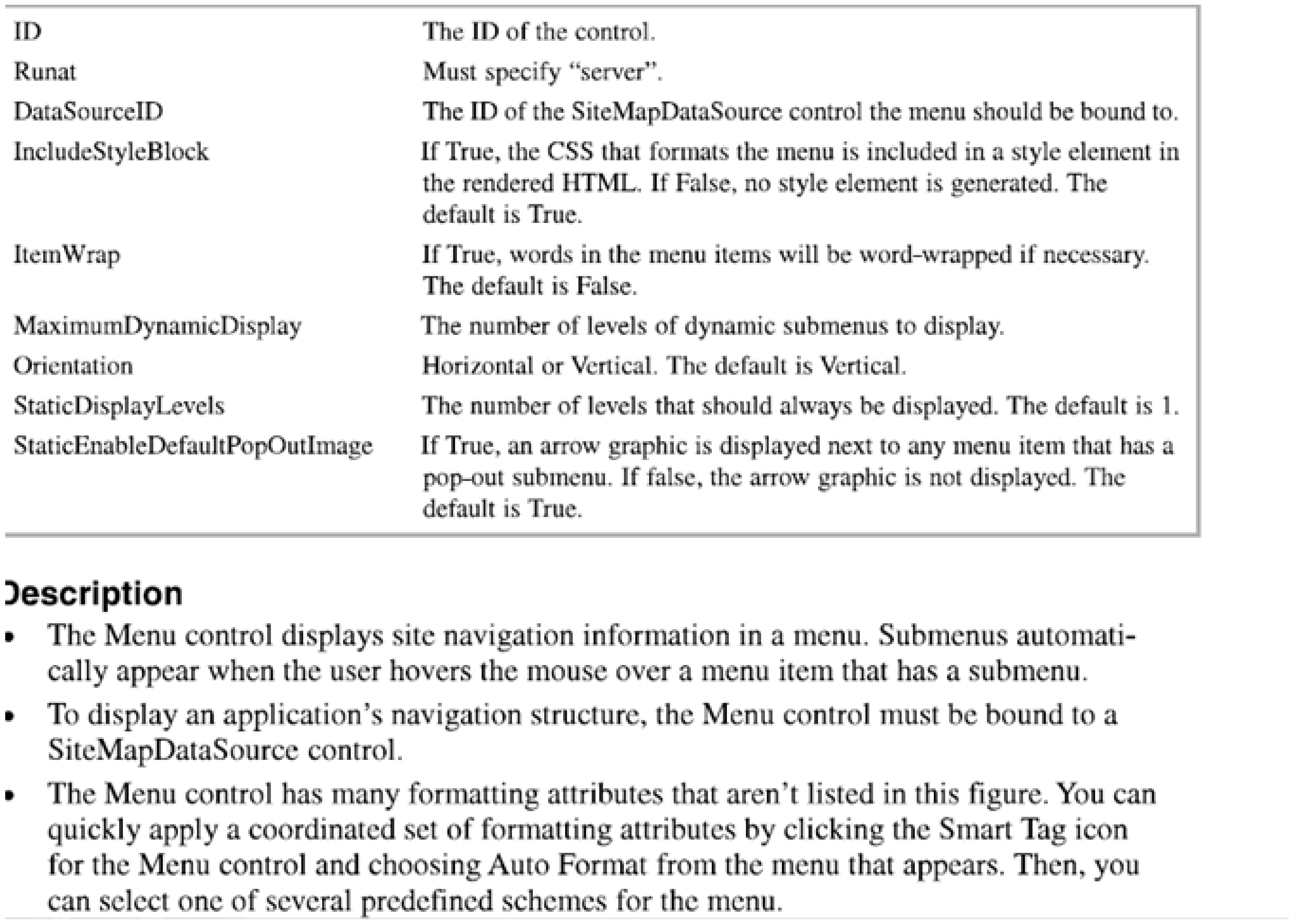
c) Explain TreeView and Menu site navigation controls.

Ans) TreeView- It provides a hierarchical view of the site structure. The user can click the + or – icon next to a node either to expand or collapse the node. Each node on the tree is a link that represents a page in a website. The Treeview control must be bound to a SiteMapDataSource control.

Attributes of the TreeView control

Attribute	Description
ID	The ID of the control.
Runat	Must specify "server".
DataSourceID	The ID of the SiteMapDataSource control the tree should be bound to.
ExpandDepth	The number of levels to be automatically expanded when the tree is initially displayed. The default is FullyExpand.
MaxDepthDataBind	Limits the maximum depth of the tree. The default is -1, which places no limit.
NodeIndent	The number of pixels to indent each level. The default is 20.
NodeWrap	Set to True to word-wrap the text of each node. The default is False.
ShowExpandCollapse	Set to False if you want to hide the Expand/Collapse buttons. The default is True.
ShowLines	Set to True to include lines that show the hierarchical structure. The default is False.

Menu Control



ID	The ID of the control.
Runat	Must specify "server".
DataSourceID	The ID of the SiteMapDataSource control the menu should be bound to.
IncludeStyleBlock	If True, the CSS that formats the menu is included in a style element in the rendered HTML. If False, no style element is generated. The default is True.
ItemWrap	If True, words in the menu items will be word-wrapped if necessary. The default is False.
MaximumDynamicDisplay	The number of levels of dynamic submenus to display.
Orientation	Horizontal or Vertical. The default is Vertical.
StaticDisplayLevels	The number of levels that should always be displayed. The default is 1.
StaticEnableDefaultPopOutImage	If True, an arrow graphic is displayed next to any menu item that has a pop-out submenu. If false, the arrow graphic is not displayed. The default is True.

Description

- The Menu control displays site navigation information in a menu. Submenus automatically appear when the user hovers the mouse over a menu item that has a submenu.
- To display an application's navigation structure, the Menu control must be bound to a SiteMapDataSource control.
- The Menu control has many formatting attributes that aren't listed in this figure. You can quickly apply a coordinated set of formatting attributes by clicking the Smart Tag icon for the Menu control and choosing Auto Format from the menu that appears. Then, you can select one of several predefined schemes for the menu.

d) What is the use of Compare Validator? Explain it along with its properties.

Ans) The compare validator compares the value entered in the control with a constant value or with the value entered in another control. You can also use the compare validator to check that the user entered a specific datatype.

Property	Description
`ValueToCompare`	The value that the control specified in the ControlToValidate property should be compared to.
`Operator`	The type of comparison to perform (Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, or DataTypeCheck).
`Type`	The data type to use for the comparison (String, Integer, Double, Date, or Currency).
`ControlToCompare`	The ID of the control that the value of the control specified in the ControlToValidate property should be compared to.

A compare validator that checks for a value greater than zero

```
<asp:TextBox ID="txtQuantity" runat="server"></asp:TextBox>&nbsp;
<asp:CompareValidator ID="CompareValidator1" runat="server"
     ControlToValidate="txtQuantity" Type="Integer"
     Operator="GreaterThan" ValueToCompare="0"
     ErrorMessage="Quantity must be greater than zero.">
</asp:CompareValidator>
```

A compare validator that checks for an integer value

```
<asp:TextBox id="txtQuantity" runat="server"></asp:TextBox>&nbsp;
<asp:CompareValidator ID="CompareValidator2" runat="server"
     ControlToValidate="txtQuantity"
     Operator="DataTypeCheck" Type="Integer"
     ErrorMessage="Quantity must be an integer.">
</asp:CompareValidator>
```

5. Attempt *<u>any two</u>* of the following:

a) Explain DetailsView Control.
Ans)

- The DetailsView control displays data for a single row of a data source. It is usually used in combination with a drop-down list or GridView control that is used to select the item to be displayed.
- The DetailsView element includes a Fields element that contains a BoundField element for each field retrieved from the data source.
- You can edit the fields collection by choosing Edit Fields from the smart tag menu of a DetailsView control.

Three modes of the DetailsView control

Mode	Description
ReadOnly	Used to display an item from the data source.
Edit	Used to edit an item in the data source.
Insert	Used to insert a new item into the data source.

DetailsView control attributes

Attribute	Description
ID	The ID of this control.
Runat	Must specify "server".
DataSourceID	The ID of the data source to bind the DetailsView control to.
DataKeyNames	A list of field names that form the primary key for the data source.
AutoGenerateRows	If True, a row is automatically generated for each field in the data source. If False, you must define the rows in the Fields element.
DefaultMode	Sets the initial mode of the DetailsView control. Valid options are Edit, Insert, or ReadOnly.
AllowPaging	Set to True to allow paging.

b) What do you mean by authentication? Explain its types.

- *Authentication* refers to the process of validating the identity of a user so the user can be granted access to an application. A user must typically supply a user name and password to be authenticated.
- After a user is authenticated, the user must still be authorized to use the requested application. The process of granting user access to an application is called *authorization.*

Windows-based authentication

- Causes the browser to display a login dialog box when the user attempts to access a restricted page.
- Is supported by most browsers.
- Is configured through the IIS management console.
- Uses Windows user accounts and directory rights to grant access to restricted pages.

Forms-based authentication

- Lets developers code a login form that gets the user name and password.
- The user name and password entered by the user are encrypted if the login page uses a secure connection.
- Doesn't rely on Windows user accounts. Instead, the application determines how to authenticate users.

Windows Live ID authentication

- *Windows Live ID* is a centralized authentication service offered by Microsoft.
- Windows Live ID lets users maintain a single user account that lets them access any web site that participates in Windows Live ID. The advantage is that the user only has to maintain one user name and password.
- To use Windows Live ID, you must register your web site with Microsoft to obtain an application ID and then download the Windows Live ID Web Authentication SDK.

c) Explain the difference between DataReader and DataAdapter in ADO.NET.

Ans) **DataReader**

- DataReader is a connection oriented architecture.
- DataReader is like a forward only recordset.
- It fetches one row at a time so very less network cost compare to DataSet(Fethces all the rows at a time).
- DataReader is readonly so we can't do any update or transaction on them.
- DataReader will be the best choice where we need to show the data to the user which requires no transaction.
- As DataReader is forward only so we can't fetch data randomly.
- .NET Data Providers optimizes the DataReader to handle huge amount of data.
- Performance is good.

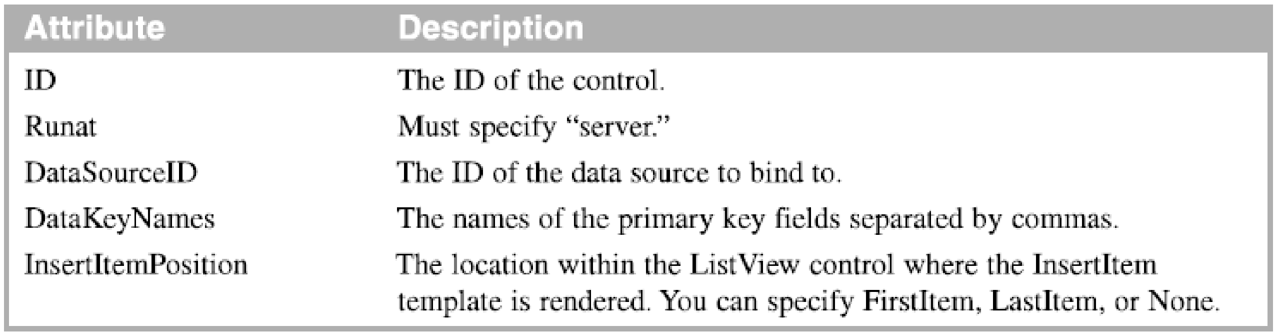
DataAdapter

- DataAdapter acts as a bridge between DataSet and database.
- DataAdapter object is used to read the data from the database and bind that data to dataset.
- DataAdapter is a disconnected oriented architecture.
- DataAdapter resolves the changes made to the DataSet back to the database.

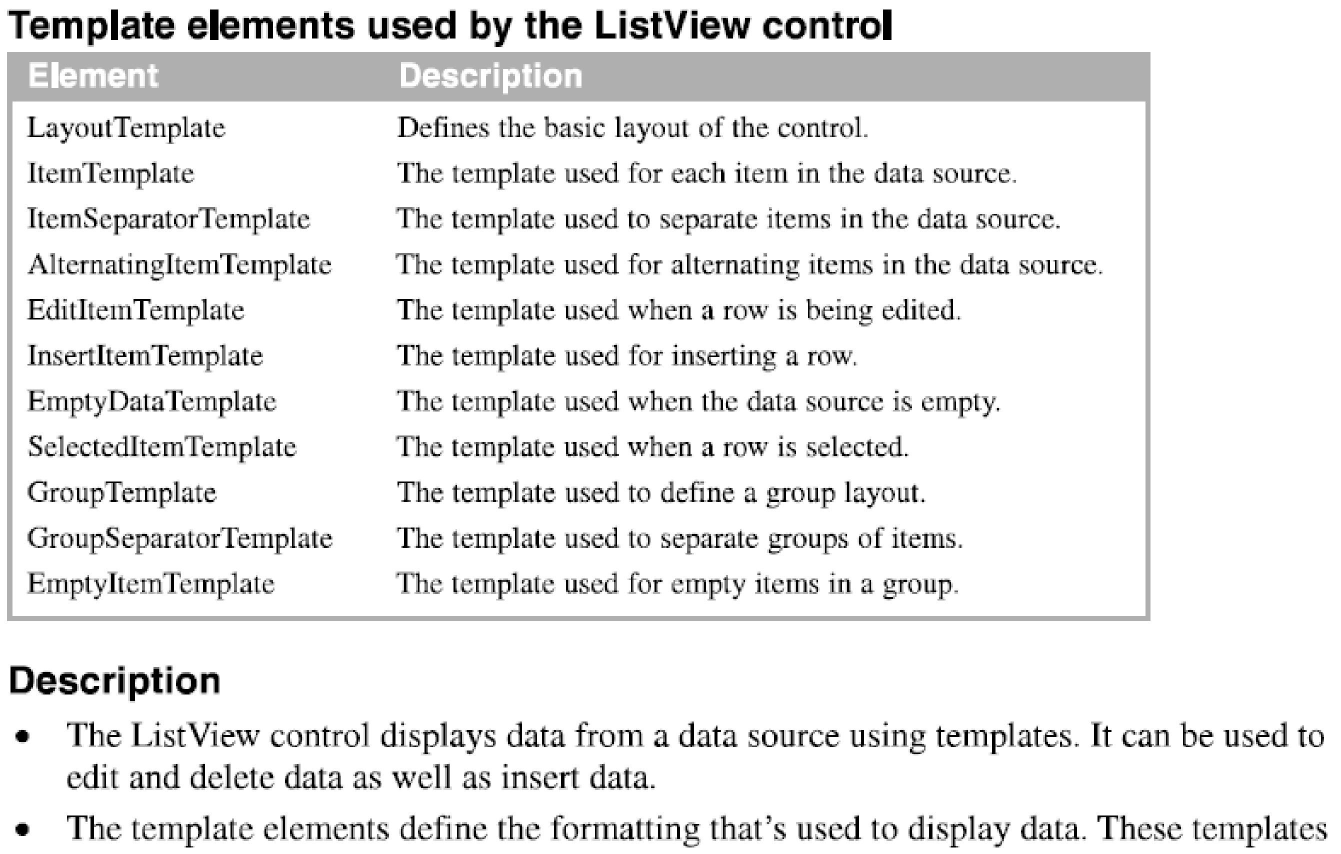
d) What is the difference between Listview and Gridview control. Explain the ListView control.

Ans) ListView presents the data in rows and columns just like a GridView control. The main difference between the ListView control and Gridview control is that the ListView control includes an additional row for inserting a new row into the table.

Basic attributes of the ListView control



Attribute	Description
ID	The ID of the control.
Runat	Must specify "server."
DataSourceID	The ID of the data source to bind to.
DataKeyNames	The names of the primary key fields separated by commas.
InsertItemPosition	The location within the ListView control where the InsertItem template is rendered. You can specify FirstItem, LastItem, or None.



Template elements used by the ListView control

Element	Description
LayoutTemplate	Defines the basic layout of the control.
ItemTemplate	The template used for each item in the data source.
ItemSeparatorTemplate	The template used to separate items in the data source.
AlternatingItemTemplate	The template used for alternating items in the data source.
EditItemTemplate	The template used when a row is being edited.
InsertItemTemplate	The template used for inserting a row.
EmptyDataTemplate	The template used when the data source is empty.
SelectedItemTemplate	The template used when a row is selected.
GroupTemplate	The template used to define a group layout.
GroupSeparatorTemplate	The template used to separate groups of items.
EmptyItemTemplate	The template used for empty items in a group.

Description

- The ListView control displays data from a data source using templates. It can be used to edit and delete data as well as insert data.
- The template elements define the formatting that's used to display data. These templates are generated automatically when you configure a ListView.

6. Attempt *any two* of the following: 10

a) Create a string array of names. Write a LINQ query to display all names from the array that contain the letter "S" and order them in ascending order. Display the query result in a Label.

Ans)

```
string[] names = new string[] { "Hanselman, Scott", "Evjen, Bill", "Haack, Phil",
                                              "Vieira,Robert", "Spaanjaars, Imar" };
var result = from name in names where author.Contains("S") orderby name select name;

foreach (var name in result)
{
Label1.Text += name + "<br />";
}
```

b) Explain the Standard Query operators "Select", "From", "orderby" and "Where" in LINQ.

Ans)

i) Select - The select keyword is used to retrieve objects from the source you are querying.

```
var allReviews = from r in myEntities.Reviews select r;
```

The r variable in this example is referred to as a *range variable* that is only available within the current query.

ii) From- It defines the collection or data source that the query must act upon.

iii) orderby - With orderby you can sort the items in the result collection. orderby is followed by an optional ascending and descending keyword to specify sort order. You can specify multiple criteria by separating them with a comma.

```
var allGenres = from g in myEntities.Genres orderby g.SortOrder descending, g.Name
select g;
```

iv) Where - The where clause in LINQ enables you to filter the objects returned by the query.

```
var allReviews = from r in myEntities.Reviews where r.Authorized == true select r;
```

c) Explain the use of UpdateProgress control in AJAX.

Ans) The Updateprogress provides status information about partial page updates in UpdatePanel controls. Despite the visual problems that postbacks usually cause, they have one big advantage: the user can see something is happening. The UpdatePanel makes this a little more difficult. Users have no visual cue that something is happening until it has happened. To tell your users to hold on for a few seconds while their request is being processed, you can use the UpdateProgress control. You usually put text such as "Please wait" or an animated image in this template to let the user know something is happening, although any other markup is acceptable as well. You connect the UpdateProgress control to an UpdatePanel using the AssociatedUpdatePanelID property. Its contents, defined in the <ProgressTemplate> element, are then displayed whenever the associated UpdatePanel is busy refreshing.

```
<asp:UpdateProgress ID=”UpdateProgress1” runat=”server”
AssociatedUpdatePanelID=”UpdatePanel1”>
<ProgressTemplate>
Please Wait an update in progress……
</ProgressTemplate>
</asp:UpdateProgress>
```

d) Write a program using jQuery that hides a paragraph on click of a button.

Ans)

```
<html><head><title></title>
<script src="Scripts/jquery-1.4.1.js" type="text/javascript"></script>
 <script type="text/javascript">
    $(document).ready(function () {
        $("p”).click(function () {
            $(this).hide();
        });
    });
   </script></head><body>
<p> ASP.NET with C# is simple </p>
</body></html>
```

7. Attempt *<u>any three</u>* of the following: 15

a) What is Method Overriding? Give an example of it.

Ans) When a method in the base class and a method in the derive class have the same name and same signature then the method in the base class overrides the method in the derive class.

The method in the base class is declared with keyword “virtual” and the method in the derive class is declared with keyword “override”.

```
using System;
class Super
{
   protected int x;
  public Super(int x)
   {
       this.x = x;
    }
public virtual void display()
{
```

```
            Console.WriteLine( “ Super x =” +x);
        }
        }
        class Sub:Super
        {
            int y;
          public Sub(int x,int y):base(x)
        {
            this.y = y;
        }
        public override void display()
        {
          Console.WriteLine( “ Super x =” + x);
          Console.WriteLine(“ Sub y = “ + y);
        }
        }
        class Overridetest
        {
        public static void Main(string[] args)
        {
          Sub s1=new Sub(100,200);
          s1.display();
        Console.ReadKey();
        }
        }
```

b) Create a delegate with two int parameters and a return type. Create a class with two delegate methods multiply and divide. Write a program to implement the delegate.

```
class Program
{
delegate double ProcessDelegate(double param1, double param2);
static double Multiply(double param1, double param2)
{
return param1 * param2;
}
static double Divide(double param1, double param2)
{
return param1 / param2;
}
static void Main(string[] args)
{
        ProcessDelegate process1= new ProcessDelegate(Multiply);
        ProcessDelegate process2= new ProcessDelegate(Divide);
        int r1=process1(8,2);
        int r2=process2(8,2);
        Console.WriteLine(“Result1 = “ + r1);
        Console.WriteLine(“Result2 = “ + r2);
        Console.ReadKey();
}
}
```

c) Give general form of switch statement. Explain with example.

Ans) The switch statement is similar to the if statement in that it executes code conditionally based on the value of a test. However, switch enables you to test for multiple values of a test variable in one go,rather than just a single condition. This test is limited to discrete values, rather than clauses such as ‘‘greater than X,’’ so its use is slightly different; but it can be a powerful technique.

The basic structure of a switch statement is as follows:

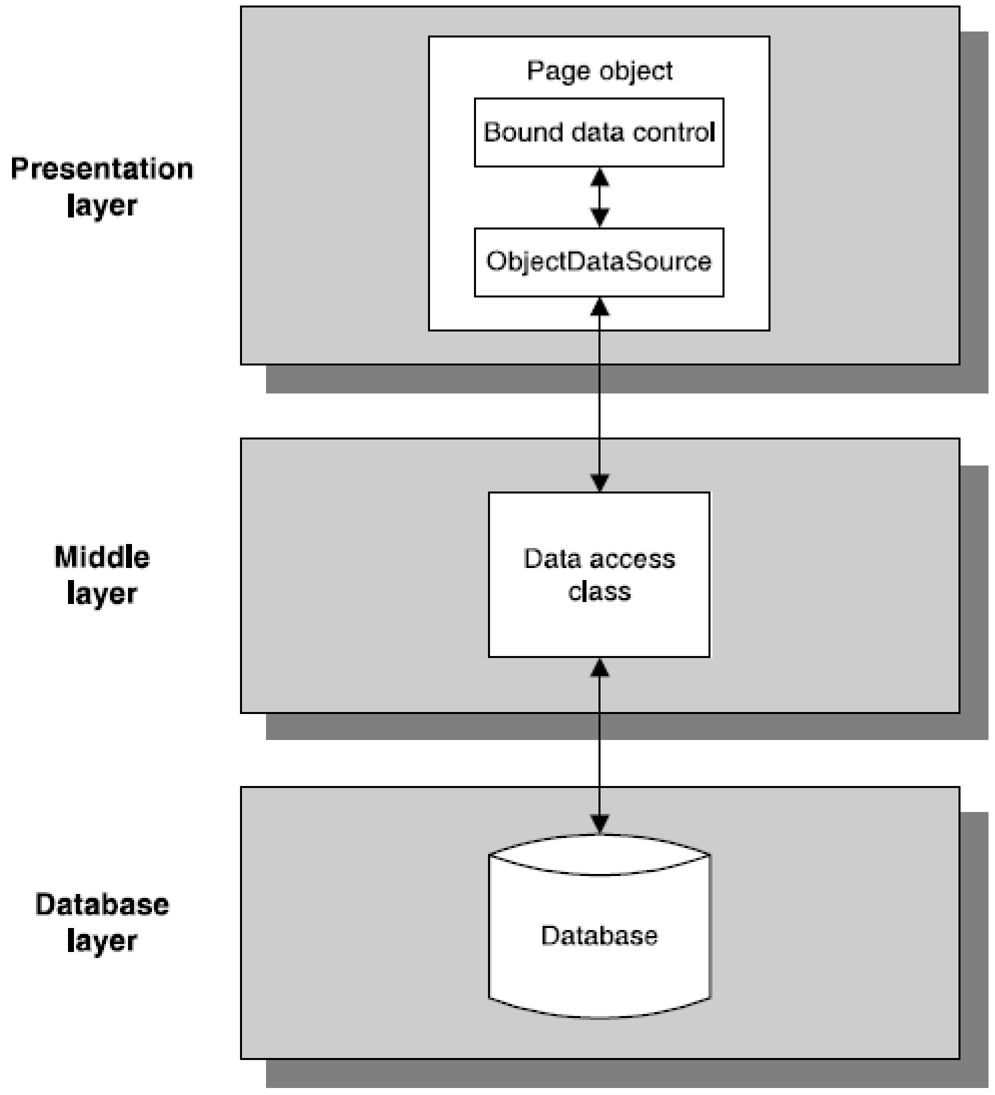
```
switch (<testVar>)
{
case <comparisonVal1>:
<code to execute if <testVar> == <comparisonVal1> >
break;
case <comparisonVal2>:
<code to execute if <testVar> == <comparisonVal2> >
break;
. . .
case <comparisonValN>:
<code to execute if <testVar> == <comparisonValN> >
break;
default:
<code to execute if <testVar> != comparisonVals>
break;
}
```

The value in <*testVar*> is compared to each of the <*comparisonValX*> values. If there is a match, then the code supplied for this match is executed. If there is no match,then the code in the default section is executed if this block exists.

```
 class SwitchEg
 {
static void Main(string[] args)
{
const string myName = "karli";
const string sexyName = "angelina";
const string sillyName = "ploppy";
string name;
Console.WriteLine("What is your name?");
name = Console.ReadLine();
switch (name.ToLower())
{
case myName:
Console.WriteLine("You have the same name as me!");
break;
case sexyName:
Console.WriteLine("My, what a sexy name you have!");
break;
case sillyName:
Console.WriteLine("That’s a very silly name.");
break;
}
Console.WriteLine("Hello {0}!", name);
Console.ReadKey();
    }}
```

d) Explain the three layer architecture of ASP.NET.

- The **presentation layer** consists of the ASP.NET pages that manage the appearance of the application. This layer can include bound data controls and ObjectDataSource objects that bind the data controls to the data.
- The **middle layer** contains the *data access classes* that manage the data access for the application. This layer can also contain business objects that represent business entities such as customers, products, or employees and that implement business rules such as credit and discount policies.
- The **database layer** consists of the database that contains the data for the application. Ideally, the SQL statements that do the database access should be saved in stored procedures within the database, but the SQL statements are often stored in the data access classes.



e) What is ViewState? How we can work with ViewState in ASP.NET.
Ans)

- *View state* is an ASP.NET feature that provides for retaining the values of page and control properties that change from one execution of a page to another.
- Before ASP.NET sends a page back to the client, it determines what changes the program has made to the properties of the page and its controls. These changes are encoded into a string that's assigned to the value of a hidden input field named _VIEWSTATE.
- When the page is posted back to the server, the _VIEWSTATE field is sent back to the server along with the HTTP request. Then, ASP.NET retrieves the property values from the _VIEWSTATE field and uses them to restore the page and control properties.
- ASP.NET also uses view state to save the values of the page properties it uses, such as IsPostBack.
- View state is *not* used to restore data entered by a user into a text box or any other input control unless the control responds to change events.
- If view state is enabled for a data-bound control, the control will not be rebound when the page is reposted. Instead, the control's values will be restored from view state.

Indexer	Description
`[name]`	The value of the view state item with the specified name. If you set the value of an item that doesn't exist, that item is created.
Property	**Description**
`Count`	The number of items in the view state collection.
`Keys`	A collection of keys for all of the items in the view state collection.
`Values`	A collection of values for all of the items in the view state collection.
Method	**Description**
`Add(name, value)`	Adds an item to the view state collection. If the item you name already exists, its value is updated.
`Clear()`	Removes all items from the view state collection.
`Remove(name)`	Removes the item with the specified name from the view state collection.

A statement that adds or updates a view state item

```
ViewState.Add("TimeStamp", Now);
```

Another way to add or update a view state item

```
ViewState["TimeStamp"] = DateTime.Now;
```

A statement that retrieves the value of a view state item

```
DateTime timeStamp = (DateTime) ViewState["TimeStamp"];
```

A statement that removes an item from view state

```
ViewState.Remove("TimeStamp");
```

f) What is AJAX? How is the processing of a web page without AJAX different from the processing of a web page with AJAX?

Ans) Ajax, which stands for **A**synchronous **J**avaScript **A**nd **X**ML, enables your client-side web pages to exchange data with the server through asynchronous calls. Probably the most popular feature driven by Ajax is the flicker-free page that enables you to perform a postback to the server without refreshing the entire page.

In traditional page processing the first drawback is because the entire page is loaded after a postback, the HTML sent to the browser is much larger than it needs to be. The second drawback of a full page reload has to do with the way the browser renders the page. Because the entire page is replaced, the browser has to dismiss the old one and then draw the new one. This causes the page to "flicker," which results in an unattractive user experience.

ASP.NET AJAX enables you to:

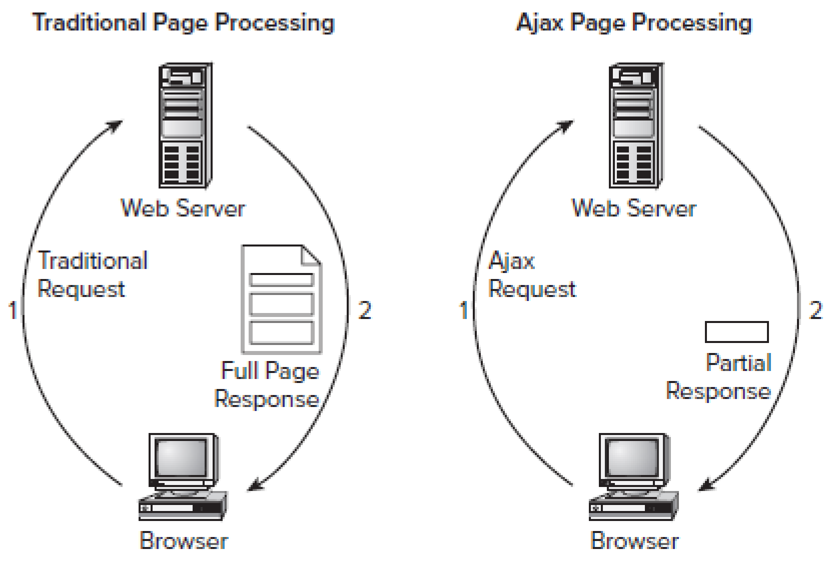
➢➢ Create flicker-free pages that enable you to refresh portions of the page without a full reload and without affecting other parts of the page

➢➢ Provide feedback to your users during these page refreshes

➢➢ Update sections of a page and call server-side code on a scheduled basis using a timer

➢➢ Access server-side web services and page methods and work with the data they return

➢➢ Use the rich, client-side programming framework to access and modify elements in your page, and get access to a code model and type system that looks similar to that of the .NET Framework.


Traditional Page Processing
Web Server
Traditional Request
1
2
Full Page Response
Browser
Ajax Page Processing
Web Server
Ajax Request
1
2
Partial Response
Browser