

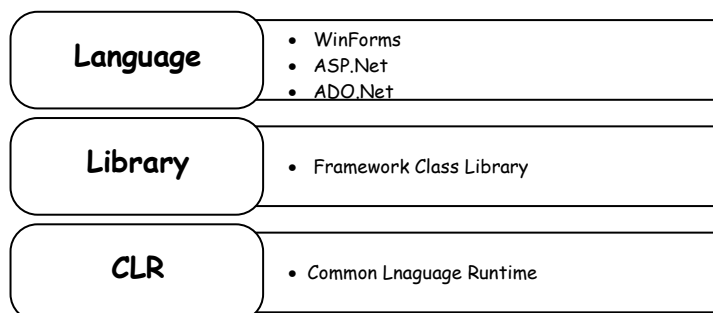
**Q.1 Attempt the following (any THREE) [15]**

**Q.1(a) Explain .Net Framework. [5]**

**Ans.:** The .Net framework is a software development platform developed by Microsoft. This framework is meant to create applications, which would run on the Windows Platform. The first version of the .Net framework was released in the year 2002. The version was called .Net framework 1.0. The .Net framework can be used to create both - **Form-based** and **Web-based** applications. Web services can also be developed using the .Net framework.

### Net Framework Architecture

The basic architecture of the .Net framework is as shown below :



**Fig. 1 : .net framework architecture diagram**

**NET Components :** .Net framework consists of following key components :

1. **Languages :** .Net supports variety of languages like Visual C++, VB, C# etc. Using these following types of applications can be built. The .Net framework Applications are classified broadly into the following categories :

- **Windows Applications :** This is used for developing Forms-based applications, which would run on an end user machine. These applications do not support remote server processing.
- **Web Applications :** The .NET Framework includes a powerful system for generating Web content dynamically, enabling personalization, security, etc. This system is called ASP.NET (Active Server Pages .NET), and one can use C# to create ASP.NET applications using Web Forms.
- **Web Services :** Using Web services one can exchange virtually any data over the Internet, using the same simple syntax regardless of the language used to create a Web service or the system on which it resides.
- **ADO.Net :** Any of the above types of Application may also require some form of database access, which can be achieved using the ADO.NET (This technology is used to develop applications to interact with Databases such as Oracle or Microsoft SQL Server).

2. **Base Class Library (BCL) :** The .NET Framework class library is a library of classes, interfaces, and value types that provide access to system functionality. It is the foundation on which .NET Framework applications, components, and controls are built. Following are different types of applications that can make use of .net class library.

- (i) Windows Application      (ii) Console Application      (iii) Web Application  
(iv) XML Web Services      (v) Windows Services

Developers just need to import the BCL in their language code and use its predefined methods and properties to implement common and complex functions like reading and writing to file, graphic rendering, database interaction, and XML document manipulation.

3. **Common Language Runtime** : The Common Language Runtime (CLR), the virtual machine component of Microsoft's .NET framework, manages the execution of .NET programs. A process known as just-in-time compiler that converts the compiled code into machine instructions which the computer's CPU then executes. The CLR provides additional services including memory management, type safety, exception handling, garbage collection, security and thread management. All programs written for the .NET framework, irrespective of programming language, are executed by the CLR.

**Q.1(b) Explain Garbage Collection in .Net.**

**[5]**

**Ans.:** In the common language runtime (CLR), the garbage collector serves as an automatic memory manager. It provides the following benefits :

- Enables one to develop an application without having to free memory.
- Allocates objects on the managed heap efficiently.
- Reclaims objects that are no longer being used, clears their memory, and keeps the memory available for future allocations. Managed objects automatically get clean content to start with, so their constructors do not have to initialize every data field.
- Provides memory safety by making sure that an object cannot use the content of another object.

**Conditions** for a garbage collection :

Garbage collection occurs when one of the following conditions is true :

- The system has low physical memory. It can be detected by either the low memory notification from the OS or low memory indicated by the host.
- The memory that is used by allocated objects on the managed heap surpasses an acceptable threshold. This threshold is continuously adjusted as the process runs.
- The `GC.Collect()` method is called to explicitly perform garbage collection. In almost all cases, there is no need to call this method, because the garbage collector runs continuously. This method is primarily used for unique situations and testing.

**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) Explain Runtime polymorphism.**

**[5]**

- Ans.:**
- When method of base class is overridden in derived class and we use base class reference to call the method then always base class version of the function is called because the compiler only see the type of reference and not the content.  
To solve this problem, the base class version of function is declared as "virtual" and the methods are overridden using "override" keyword.
  - Now depending on object assigned to base class reference, respective version of function is called.
  - Hence the polymorphism is resolved at runtime, hence known as runtime. Polymorphism also known as late binding/ dynamic binding.

```
using System;
namespace inheritance1
{
    class A
    {
        public virtual void display()
        {
            Console.WriteLine("class A display");
        }
    }

    class B : A
    {
        public override void display()
        {
            Console.WriteLine("class B display");
        }
    }

    class C : A
    {
        public override void display()
        {
            Console.WriteLine("class C display");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
```

```

        A r; //reference of A

        r = new A();
        r.display();

        r = new B();
        r.display();
        r = new C();
        r.display();

        Console.ReadLine();
    }
}

class Program
{
    static void Main (String [ ] args)
    {
        A a1 = new A ( );
        B b1 = new B ( );
        C c1 = new C ( );

        A r; // reference of A
        r = a1;
        r : display ( );

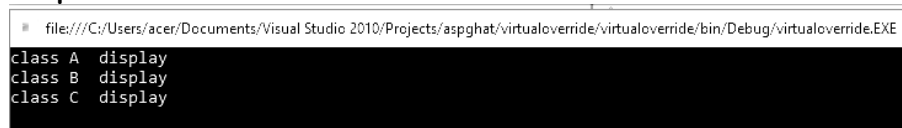
        r = b1;
        r : display ( );

        r = c1;
        r : display ( );

        C.RL ( );
    }
}

```

**Output:**



```

file:///C:/Users/acer/Documents/Visual Studio 2010/Projects/aspghat/virtualoverride/virtualoverride/bin/Debug/virtualoverride.EXE
class A display
class B display
class C display

```

**Q.1(e) Explain Assembly and its component.**

**[5]**

- Ans.:**
- An assembly is a fundamental building block of any .NET framework application. It contains the code that is executed by common language runtime. For example, when we build a simple C# application, Visual Studio creates an assembly in the form of a single portable executable (PE) file, specifically an EXE or DLL.
  - Every assembly has a file called 'manifest' file that stores all information about that assembly. This information's are known as metadata.
  - The manifest file contains all the metadata needed to specify the assembly's version requirements, security identity, and all metadata needed to define the scope of the assembly and resolve references to resources and classes.

**Components of Assembly :**

- **Manifest**  
It describes the assembly. The manifest file contains all the metadata needed to specify the assembly's version requirements, security identity, and all metadata needed to define the scope of the assembly and resolve references to resources and classes.
- **Type Metadata**  
It contains metadata, which describes each and every type (class, structure, enumeration, etc).
- **MSIL**  
It contains Intermediate language code.
- **Resources**  
It contains bitmaps, icons, audios and other types of resources.

**Q.1(f) Explain Namespace in C#.****[5]**

**Ans.:** A namespace is designed for providing a way to keep one set of names separate from another. The class names declared in one namespace does not conflict with the same class names declared in another.

**Defining a Namespace:**

A namespace definition begins with the keyword namespace followed by the namespace name as follows :

```
namespace namespace_name
{
    // code declarations
}
```

To call the namespace-enabled version of either function or variable, prepend the namespace name as follows :

```
namespace_name.item_name;
```

**Namespaces have the following properties:**

- They organize large code projects.
- They are delimited by using the . operator.
- The using directive obviates the requirement to specify the name of the namespace for every class.
- The global namespace is the "root" namespace: global::System will always refer to the .NET Framework namespace System.
- Namespaces can be nested within other Namespaces.

**Example:**

```
using System;
namespace nestednamespace
{
    namespace n1
    {
        public class A
        {
            public void f1()
            {
                Console.WriteLine("f1 of A of n1");
            }
        }
    }
    namespace n2
    {
```

```

public class A
{
    public void f1()
    {
        Console.WriteLine("f1 of A of n2");
    }
}
class program
{
    static void Main(string[] args)
    {
        n2.A a1 = new n2.A();
        a1.f1();
        n1.A a2 = new n1.A();
        a2.f1();
        Console.ReadLine();
    }
}
}

```

**Output:**



**Q.2 Attempt the following (any THREE)**

**[15]**

**Q.2(a) Explain ASP.NET File Types.**

**[5]**

**Ans.:** ASP.NET applications can include many types of files. They are as follows :

1	Ends with .aspx	These are ASP.NET web pages. They contain the user interface and, optionally, the underlying application code. Users request or navigate directly to one of these pages to start your web application.
2	Ends with .ascx	These are ASP.NET user controls. User controls are similar to web pages, except that the user can't access these files directly. Instead, they must be hosted inside an ASP.NET web page. User controls allow you to develop a small piece of user interface and reuse it in as many web forms as you want without repetitive code.
3	web.config	This is the configuration file for your ASP.NET application. It includes settings for customizing security, state management, memory management, and much more.
4	global.asax	This is the global application file. You can use this file to define global variables (variables that can be accessed from any web page in the web application) and react to global events (such as when a web application first starts).
5	Ends with .cs	These are code-behind files that contain C# code. They allow you to separate the application logic from the user interface of a web page.

**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) Explain : (i) RangeValidator Control in ASP.Net [5]**  
**(ii) CustomValidator Control in ASP.Net**

**Ans.: (i) RangeValidator Control in ASP.Net**

This control is used to compare the value of one input control to the value of another input control (Generally the text control) or to a fixed value. The control has ControlToCompare & ControlToValidate properties for the controls to compare and validate. The control has the ErrorMessage property to show the validation summary if validation fails. It has the special property called as operator to determine the type of comparison between the controls; it accepts the values like "GreaterThan", "Equal", "LessThan" etc. The type property is used to specify the data type of the values to compare.

**Example:**

```
<asp:TextBox id="t1" runat="server" />
<asp:TextBox id="t2" runat="server" />
<asp:CompareValidator id="compare" ControlToValidate="txt1"
ControlToCompare="t2" ForeColor="Pink" BackColor="maroon" Type="String"
Text="Validation Failed!" runat="server"/>
```

**(ii) CustomValidator Control in ASP.Net**

This control allows us to write a method to handle the validation of the value entered by the user. The control has ClientValidation Function property which specifies the name of the client-side validation script function to be executed. It also has the ControlToValidate, ErrorMessage, text, id etc. properties as explained in earlier controls.

**Example:**

```
<asp:Label runat="server" Text="Enter a username: "/>
<asp:TextBox id="t1" runat="server"/>
<asp:Button Text="Submit" runat="server"/><br/>
<asp:Label id="msg" runat="server"/><br/>
<asp:CustomValidator ControlToValidate="t1"
OnServerValidate="user" Text="Enter proper value!" runat="server"/>
```

**Q.2(d) Explain AdRotator Control in Asp.Net. [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.
  - 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 *AdRotator* provides following features:

**The Advertisement File :**

Syntax:

```
<Advertisements>
```

```
<Ad>
<!-- First ad here. -->
</Ad>
<Ad>
<!-- Second ad here. -->
</Ad>
</Advertisements>
```

**Advertisement File Elements:**

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 some newer browsers.
Impressions	A number that sets how often an advertisement will appear. This number is relative to the numbers specified for other ads
Keyword	A keyword that identifies a group of advertisements. You can use this for filtering

**The AdRotator Class :**

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).

**Special Frame Targets :**

Target	Description
_blank	The link opens a new unframed window.
_parent	The link opens in the parent of the current frame.
_self	The link opens in the current frame.
_top	The link opens in the topmost frame of the current window

**Q.2(e) Explain SiteMapPath in Asp.Net.**

**[5]**

**Ans.:** The Sitemap Path control displays the navigation path of the current page. The path acts as clickable links to previous pages. The Sitemap Path control uses the web.sitemap file by default. The SiteMapPath control creates navigation mechanism which is a linear path defining where the user is currently located in navigation arrangement. It usually helps end user to know his location in relation to the rest of the site.

**Following are some important properties of SiteMapPath control :**

Properties	Description
PathSeparator	It is to get or set the Path separator text. (By default it is >.)
PathDirection	It is to set the direction of the links generated in the output. Possible values for this property: RootToCurrent and CurrentToRoot.
NodeStyle	It is to set the style of all nodes that will be displayed.
CurrentNodeStyle	It is to set the style on node that represent the current page.
RootNodeStyle	It is to set the style on the absolute root node.
PathSeparatorStyle	It is to set the style of path separator.
ShowToolTips	It is to set the tooltip for the control. By default value is true.



To display SiteMapPath control in all your pages automatically, add it in the master page of the website. To Create a SiteMapPath Control follow the following steps :

- (i) Open the master page in design View and drag a SiteMapPath control from the Navigation Toolbox to the page.

```
<asp:SiteMapPath ID="SiteMapPath1" runat="server">
</asp:SiteMapPath>
```

- (ii) Save the changes and then request any Content page in the browser.

- (iii) Content page shows the path from the root of the site to the current page. While navigating through the site the SiteMapPath control changes the path for each page.

**Q.2(f) Explain Different methods to perform Redirection in Code.**

**[5]**

- Ans.:**
- To perform redirection in code, you first need a control that causes the page to be posted back, i.e you need an event handler that reacts to the ServerClick event of a control such as HtmlInputButton or HtmlAnchor.
  - When the page is posted back and your event handler runs, you can use the HttpResponse.Redirect() method to send the user to the new page. You can get access to the current HttpResponse object through the Page.Response property.
  - When you use the Redirect() method, ASP.NET immediately stops processing the page and sends a redirect message back to the browser. Any code that occurs after the Redirect() call won't be executed.
  - When the browser receives the redirect message, it sends a request for the new page. You can use the Redirect() method to send the user to any type of page. You can even send the user to another website by using an absolute URL (a URL that starts with http://), e.g., Response.Redirect ("http://www.abc.com");
  - ASP.NET gives you one other option for sending the user to a new page. You can use the HttpServerUtility.Transfer() method instead of Response.Redirect(). An HttpServerUtility object is provided through the Page.Server property, so your redirection code would look like this: Server.Transfer ("newpage.aspx");
  - The advantage of using the Transfer() method is that it doesn't involve the browser. Instead of sending a redirect message back to the browser, ASP.NET simply starts processing the new page as though the user had originally requested that page. This behavior saves a bit of time, but it also introduces some significant limitations.
  - You can't use Transfer() to send the user to another website or to a non-ASP.NET page (such as an HTML page). The Transfer() method allows you to jump only from one ASP.NET page to another, in the same web application.
  - When you use Transfer(), the user won't have any idea that another page has taken over, because the browser will still show the original URL. This can cause a problem if you want to support browser bookmarks. It's much more common to use HttpResponse.Redirect() than HttpServerUtility.Transfer().

**Q.3 Attempt the following (any THREE)**

**[15]**

**Q.3(a) Explain Exception Handling.**

**[5]**

**Ans.:** Exception Handling :

- Exceptions are problems encountered in program at run time.
- C# supports exception handling.
- When an exception occurs in code notification is sent, and exception handling code are specified to handle exception in rational manner.
- General Structure of Exception Handling Mechanism :

```
try
{
    // Risky code goes here (opening a file, connecting to a database, and so on).
}
catch
{
```

```
// An error has been detected. You can deal with it here.
}
finally
{
    // Time to clean up, regardless of whether or not there was an error.
}
```

- To handle exception C# uses following keywords.

**(i) try :**

- ▷ It contains block of code that needs to be monitored for occurrence of exception.
- ▷ If exception occurs within try block there is a chance of it being handled.
- ▷ try block can be followed by either catch or finally or both.

**(ii) catch :**

- ▷ It contains exception handling code.

**(iii) throw :**

- ▷ used to throw exception manually.
- ▷ Mainly used for throwing user defined exception.

**(iv) finally :**

- ▷ Contains code that needs to be executed irrespective. Exception is generated or not. Generated exception is handled or not.

**Q.3(b) Explain View State in Asp.Net.**

**[5]**

**Ans.:**

- View State is one of the most common ways to store information.
- View state uses a hidden field that ASP.NET automatically inserts in the final, rendered HTML of a web page.
- View State is a perfect place to store information that's used for multiple post backs in a single web page.
- The **ViewState Collection** :
  - i) The ViewState property of the page provides the current view-state information.
  - ii) This property provides an instance of the StateBag collection class.
  - iii) The StateBag is a dictionary collection, which means every item is stored in a separate "slot" using a unique string name, which is also called the key name.
  - iv) Example :

```
// The this keyword refers to the current Page object. It's optional.
    this.ViewState["Counter"] = 1;
```
  - v) This places the value 1 (or rather, an integer that contains the value 1) into the ViewState collection and gives it the descriptive name Counter.
  - vi) If currently no item has the name Counter, a new item will be added automatically. If an item is already stored under the name Counter, it will be replaced.
  - vii) When retrieving a value, you use the key name.
  - viii) Here's the code that retrieves the counter from view state and converts it to an integer:

```
int counter;
counter = (int)this.ViewState["Counter"];
```

**Q.3(c) Explain how state is managed using cookies.**

**[5]**

**Ans.:**

- Cookies provide another way to store state information for later use.
- Cookies are small files that are created in the web browser's memory or on the client's hard drive.
- One advantage of cookies is that they work transparently, without the user being aware that information needs to be stored.
- They also can be easily used by any page in your application and even be retained between visits, which allows for truly long-term storage.
- Cookies are easily accessible and readable if the user finds and opens the Corresponding file.
- Some users disable cookies on their browsers, which will cause problems for web applications that require them.
- **Steps to use cookies :**
  - (i) One should import the System.Net namespace so you can easily work with the appropriate types:  
using System.Net;
  - (ii) To set a cookie, just create a new HttpCookie object. It can be then filled with string information and can be attached to the current web response:  
// Create the cookie object.  
HttpCookie cookie = new HttpCookie("Priority");  
// Set a value in it.  
cookie["SubjectPriority"] = "ASP";  
// Add another value.  
cookie["Country"] = "INDIA";  
// Add it to the current web response.  
Response.Cookies.Add(cookie);
  - (iii) A cookie added in this way will persist until the user closes the browser and will be sent with every request. To create a longer-lived cookie, you can set an expiration date:  
// This cookie lives for one year.  
cookie.Expires = DateTime.Now.AddYears(1);
  - (iv) You retrieve cookies by cookie name, using the Request.Cookies collection.  
HttpCookie cookie = Request.Cookies["Priority"];
  - (v) The only way to remove a cookie is by replacing it with a cookie that has an expiration date that has already passed. This code demonstrates the technique:  
HttpCookie cookie = new HttpCookie("Priority");  
cookie.Expires = DateTime.Now.AddDays(-1);  
Response.Cookies.Add(cookie);

**Q.3(d) Compare Session State and Application State Management techniques.**

**[5]**

**Ans.:**

	<b>Session State</b>	<b>Application State</b>
1.	It allows all .NET data types for the default in-process storage mode. All serializable .NET data types if you use an out-of-process storage mode.	It Allows all .NET data types
2.	The storage location is Server memory, state service, or SQL Server, depending on the mode you choose.	The storage location is Server memory.
3.	Times out after a predefined period (usually 20 minutes, but can be altered globally or programmatically).	The lifetime of the application (typically, until the server is rebooted).
4.	The scope is the whole ASP.NET application.	The scope is the whole ASP.NET application. Unlike other methods, application data is global to all users.

5.	It is very secure, because data is never transmitted to the client	It is very secure, because data is never transmitted to the client.
6.	The performance is slow when storing a large amount of information, especially if there are many users at once, because each user will have their own copy of session data.	The performance is slow when storing a large amount of information, because this data will never time out and be removed.
7.	Typical used for Storing items in a shopping basket	Typical used for storing any type of global data

**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 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, 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 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, 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 <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>

#### 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) Explain relation between master page and content page.

[5]

**Ans.:** Master page provides a framework within which the content from other pages can be displayed.

- It provides elements such as headers, footers, style definitions, or navigation bars that are common to all pages in your web site. So the Content Pages need not have to duplicate code for shared elements within your Web site.
- It gives a consistent look and feel for all pages in your application.
- The master page layout consists of regions where the content from each content page should be displayed.
- These regions can be set using ContentPlaceHolder server controls.
- These are the regions where you are going to have dynamic content in your page.
- A derived page also known as a content page is simply a collection of blocks the runtime will use to fill the regions in the master.
- To provide content for a ContentPlaceHolder, you use another specialized control, called Content.
- The ContentPlaceHolder control and the Content control have a one-to-one relationship.
- For each ContentPlaceHolder in the master page, the content page supplies a matching Content control.
- ASP.NET links the Content control to the appropriate ContentPlaceHolder by matching the ID of the ContentPlaceHolder with the Content ContentPlaceHolderID property of the corresponding Content control.

```

MasterPage.master x Default.aspx*
Client Objects & Events (No Events)
<asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
</asp:ContentPlaceHolder>
Default.aspx x MasterPage.master
Client Objects & Events (No Events)
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
</asp:Content>
    
```

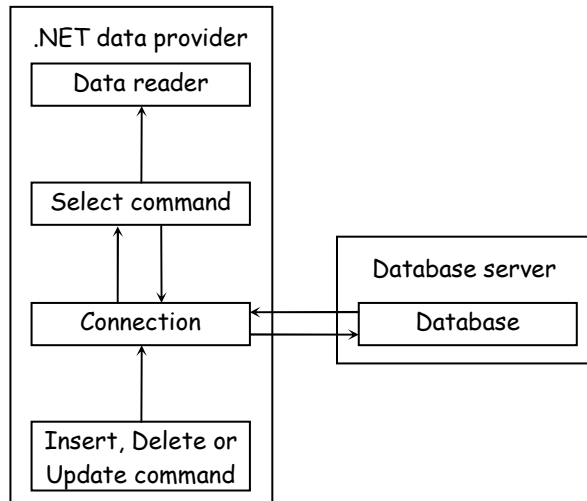
Q.4 Attempt the following (any THREE)

[15]

Q.4(a) Explain in details connected Mode in ADO.NET.

[5]

Ans.: ADO.NET components for accessing a database directly :



- The *connected* environment provides forward-only, read-only access to data in the data source and the ability to execute commands against the data source. The connected classes provide a common way to work with connected data regardless of the underlying data source.
- They include Connection, Command, DataReader, Transaction, ParameterCollection, and Parameter classes.
- **Connected Mode Classes :**  
The following classes are used by ADO.NET to communicate directly with the data source :
  - **Connection** : Maintains information required to connect to the data source through a connection string. The connection string contains information such as the name of the data source and its location, and authorization credentials and settings.
  - **Command** : Executes SQL statements or stored procedures against the data source. The command class has a ParameterCollection object containing Parameter objects that allow parameterized SQL statements and stored procedures to be used against the data source.
  - **DataReader** : Provides connected forward-only, read-only access to the data source. It is optimized for speed.
  - **Parameter** : Allows parameters for both parameterized queries and stored procedures to be defined and set to appropriate values. The Parameter class is accessed through the ParametersCollection object within a Command object. It supports input and output parameters as well as return values from stored procedures.
  - **Transaction** : Allows transactions to be created on a connection so that multiple changes to data in a data source are treated as a single unit of work and either all committed or cancelled.

**Q.4(b) Compare DataReader and DataAdapter.**

**[5]**

Ans. :

	<b>DataReader</b>	<b>DataAdapter</b>
1	The ADO.NET DataReader is used to retrieve read-only (cannot update data back to datasources) and forward-only (cannot read backward/random) data from a database.	The DataSet is an in-memory representation of data. The DataSet represents a complete set of data including related tables, constraints, and relationships among the tables.
2	Using the DataReader increases application performance and reduces system overheads. This is due to one row at a time is stored in memory.	Using the DataAdapter decreases application performance and increases system overheads.
3	You create a DataReader by calling <code>Command.ExecuteReader()</code> after creating an instance of the <code>Command</code> object.	Object is created explicitly using syntax: <code>SqlDataAdapter myAdapter = new SqlDataAdapter(SQLquery, conn);</code>
4	This is a connected architecture: The data is available as long as the connection with database exists.	This works for Disconnected Architecture.
5	You need to open and close the connection manually in code.	No need to manually open and close connection in code.

**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 :

<b>Property</b>	<b>Description</b>
Connection	The <code>SqlConnection</code> 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 <code>CommandType</code> enumeration that indicates whether the <code>CommandText</code> property contains a SQL statement ( <code>Text</code> ) or the name of a stored procedure ( <code>StoredProcedure</code> ).
Parameters	The collection of parameters used by the command.

Methods of `SqlCommand` are :

<b>Method</b>	<b>Description</b>
<code>ExecuteReader</code>	Executes a query and returns the result as a <code>SqlDataReader</code> object.
<code>ExecuteNonQuery</code>	Executes the command and returns an integer representing the number of rows affected.
<code>ExecuteScalar</code>	Executes a query and returns the first column of the first row returned by the query.

**Q.4(d) Explain how Repeated-Value Data Binding work?**

**[5]**

Ans. :

- Repeated-value data binding works with the ASP.NET list controls.
- To use repeated-value binding, you link one of these controls to a data source.
- When you call `DataBind()`, the control automatically creates a full list by using all the corresponding values.

- Repeated-value binding can also simplify your life by supporting advanced formatting and template options that automatically configure how the data should look when it's placed in the control.
- To create a data expression for list binding, you need to use a list control that explicitly supports data binding.
- ASP.NET provides a number of list controls :
  - ▷ **ListBox, DropDownList, CheckBoxLayout, and RadioButtonList** : These web controls provide a list for a single field of information.
  - ▷ **HtmlSelect** : This server-side HTML control represents the HTML <select> element and works essentially the same way as the ListBox web control.
  - ▷ **GridView, DetailsView, FormView, and ListView** : These rich web controls allow you to provide repeating lists or grids that can display more than one field of information at a time.

**Q.4(e) Explain various data source controls in ADO.NET.**

**[5]**

**Ans.:** Following are the various data source controls in ADO.NET :

- **SqlDataSource** : This data source allows you to connect to any data source that has an ADO.NET data provider. This includes SQL Server, Oracle, and OLE DB or ODBC data sources. When using this data source, you don't need to write the data access code.
- **AccessDataSource** : This data source allows you to read and write the data in an Access database file (.mdb). However, its use is discouraged, because Access doesn't scale well to large numbers of users.
- **ObjectDataSource** : This data source allows you to connect to a custom data access class. This is the preferred approach for large-scale professional web applications, but it forces you to write much more code.
- **XmlDataSource** : This data source allows you to connect to an XML file.
- **SiteMapDataSource** : This data source allows you to connect to a .sitemap file that describes the navigational structure of your website.
- **EntityDataSource** : This data source allows you to query a database by using the LINQ to Entities feature.
- **LinqDataSource** : This data source allows you to query a database by using the LINQ to SQL feature, which is a similar (but somewhat less powerful) predecessor to LINQ to Entities.

**Q.4(f) What is a GridView control?**

**[5]**

**Ans.:** The GridView server control displays data provided by a data source in a tabular format. It renders its data as an HTML table. The GridView control supports automatic paging, sorting, editing, deleting, and selecting. To enable row selection *AutoGenerateSelectButton* property of GridView can be set to true.

The GridView data can be sorted based on any or all of its columns. To sort GridView data, following are the steps :

- Set **AllowSorting** attribute of GridView to True.
- Also set **SortExpression** property of columns to respective field from database to sort.

Paging refers to the ability of GridView control to display bound data one page at a time. Users can arbitrarily select pages from a GridView. To enable paging feature of a GridView :

- Set **AllowPaging** property of GridView control to true.
- Set **PageSize** property to no of records we want in each page.

**Example :**

```
<asp:gridview AllowSorting="true" AllowPaging="true" PageSize="5" ID="Gridview1"
runat="server" DataKeyNames="pid" DataSourceID="SqlDS" >
<Columns>
```



```
<asp:BoundField DataField="pname" HeaderText="PRODUCT NAME"
SortExpression="pname"> </asp:BoundField>
</Columns>
</asp:gridview>
```

**Basic attributes of the GridView 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.
AutoGenerateRows	If True, a row is automatically generated foreach 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.

**Q.5 Attempt the following (any THREE)**

[15]

**Q.5(a) Explain XML Schema definition.**

[5]

- Ans.:
- An XSD, or schema, defines what elements and attributes a document should contain and the way these nodes are organized (the structure).
  - It can also identify the appropriate data types for all the content.
  - Schema documents are written using an XML syntax with specific element names.
  - All the XSD elements are placed in the <http://www.w3.org/2001/XMLSchema> namespace. Often, this namespace uses the prefix `xsd:` or `xs:`.

- Here is a simple XSD code.

```
<?xml version="1.0"?>
<xs:schema
targetNamespace="http://www.SuperProProducts.com/SuperProProductList"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" >
<xs:element name="SuperProProductList">
<xs:complexType>
<xs:sequence maxOccurs="unbounded">
<xs:element name="Product">
<xs:complexType>
<xs:sequence>
<xs:element name="Price" Type="xs:double" />
</xs:sequence>
<xs:attribute name="ID" use="required" Type="xs:int" />
<xs:attribute name="Name" use="required" Type="xs:string" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

**Features :**

- XSDs can be extensible for future additions.
- XSD is richer and more powerful than DTD.
- XSD is written in XML.
- XSD supports data types.

- XSD supports namespaces.
- XSD is W3C recommendation.
- An XML XSD is kept in a separate document and then the document can be linked to an XML document to use it.

**Q.5(b) Explain Windows authentication in ASP.Net.**

**[5]**

**Ans.:** The Windows authentication provider is the default provider for ASP.NET. It authenticates the users based on the users Windows accounts. Windows authentication in ASP.NET actually relies on IIS to do the authentication. IIS can be configured so that only users on a Windows domain can log in. If a user attempts to access a page and is not authenticated then user will be shown a dialog box asking them to enter their username and password. This information is then passed to the Web server and checked against the list of users in the domain. If the user has supplied valid credentials access is granted.

To configure ASP.NET to use Windows authentication use the following code :

```
<system.web>
<authentication mode="Windows"/>
<authorization>
<allow users = "*">
</authorization>
</system.web>
<allow users="*" />
```

statement specifies permissions are provided to authorized users as well anonymous users.

There are four different kinds of Windows authentication options available that can be configured in IIS :

- **Anonymous Authentication** : In this, IIS does not perform any authentication check and allows access to any user to the ASP.NET application.
- **Basic Authentication** : In this, a Windows user name and password have to be provided to connect. This information is sent over the network in plain text and hence this is an insecure kind of authentication.
- **Digest Authentication** : It is almost same as Basic Authentication but the password is hashed before it is sent across the network.
- **Integrated Windows Authentication** : In this, password is not sent across the network and some protocols are used to authenticate users. It provides the tools for authentication and strong cryptography is used to help to secure information in systems across entire network.

**Q.5(c) Explain Authorization in Asp.Net.**

**[5]**

**Ans.:** Authentication and Authorization are two interconnected security concepts. Authentication is a process of identifying a user and authorization is the process of checking whether authenticated user has access to the resources they requested.

There are two forms of authorization available in ASP.NET :

- **FileAuthorization** : It is performed by the FileAuthorizationModule. It uses the access control list (ACL) of the .aspx file to resolve whether a user should have access to the file. ACL permissions are confirmed for the users Windows identity.
- **UrlAuthorization**: In the Web.config file you can specify the authorization rules for various directories or files using the<authorization> element.

Following is the syntax Authentication :

```
<authorization>
<[allow | deny] users roles />
</authorization>
```

The allow and deny tags specifies grant or revoke respectively.

Example :

```
<system.web>
<authorization>
<allow users = "user1" />
<deny users="*" />
</authorization>
</system.web>
```

It will allow only user user1 and deny all other users to access that application. If you want to give permission for more users then just add usernames separated with comma like user2,user3 etc.

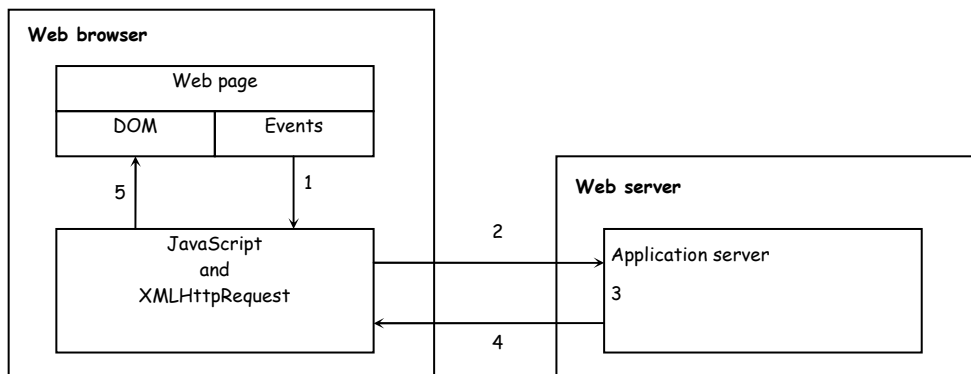
If you want to allow only admin roles to access the application and deny permission for all the roles, then write the following code in web.config.

```
<system.web>
<Authorization>
<allow roles="Admin" />
<deny users="*" />
</Authorization>
</system.web>
```

**Q.5(d) Explain the architecture of AJAX.**

**[5]**

**Ans.:** The architecture of AJAX is as follows :



AJAX updates the contents of a web page as following :

1. An event happens on the web page. This can be the user moving the mouse, clicking a button etc. This event triggers JavaScript code to execute.
2. JavaScript prepares a request and sends it to the web server. The request contains information about the event and the current state of the controls on the web page.
3. The server receives the data and processes it. Although processing can take place on the client, some actions, such as database access, must happen on the server.
4. The server prepares a response and sends it back to the browser. The response contains the updated state of the controls on the web page.
5. JavaScript parses the response and uses the data it contains to update the contents of the web page by modifying objects in the DOM. The browser then updates the user's screen.

**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 containsPostBackTrigger and AsyncPostBackTrigger elements.
RenderMode	This property can be set to Block or Inline to indicate whether the UpdatePanel renders itself as a <div> or <span> 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 is 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 UpdateProgress Control in Asp.Net.**

**[5]**

- Ans.:**
- Even though there was visual problem in postback but it has one advantage that user can see something is happening however with update panel user will have no idea that something is happening until it has happened.
  - To tell user to wait for few seconds while their request is processed, we can use update progress control.
  - The UpdateProgress control allows you to show a message while a time-consuming update is under way.
  - **Showing a Simulated Progress Bar :**  
When you add the UpdateProgress control to a page, you get the ability to specify some content that will appear as soon as an asynchronous request is started and will disappear as soon as the request is finished.

**Properties :**

- DisplayAfter*: It determines the time in milliseconds that the control has to wait before it displays its content.
- DynamicalLayout*: It determine whether the progress template is dynamically rendered.
- Visible*: Indicated whether the control is visible and rendered.
- EnableViewState* : Whether the control automatically saves its state for the use in round trips.

**General Form:**

```
<asp:UpdateProgress ID="UpdateProgress1" runat="server">
</asp:UpdateProgress>
```

