(2½ hours)

Total Marks: 75

N. B.: (1) **All** questions are **compulsory**.
(2) Make **suitable assumptions** wherever necessary and **state the assumptions** made.
(3) Answers to the **same question** must be **written together**.
(4) Numbers to the **right** indicate **marks**.
(5) Draw **neat labeled diagrams** wherever **necessary**.
(6) Use of **Non-programmable** calculators is **allowed**.

1.	**Attempt *any three* of the following:**	**15**
a.	What are .NET languages? Explain various features of C# languages. The Microsoft .Net Framework provides the necessary foundation for developing windows as well as web applications. The unique feature of .NET is the multi-language support that it provides. To help create languages for the .NET Framework, Microsoft created the Common Language Infrastructure specification (CLI). The CLI describes the features that each language must provide in order to use the .NET Framework and common language runtime. The .NET platform supports many languages and the list is growing day by day. The C# compiler is considered to be the most efficient compiler in the .NET family. Features of C#: --modern—type safe—inter operable--object oriented--easy to learn—etc.	
b.	What is property? Explain read-write property with proper syntax and example.	

c.	Explain static constructor and copy constructor with examples.
	
	Static Constructor
	A static constructor is used to initialize static variables of a class. It is declared using the static keyword. A class can have only one static constructor. The general syntax of a static constructor is shown below.
	
	static classname()
	{
	static member initializations;
	}
	Also note that a static constructor is parameter less and no access modifiers are allowed with its declaration. The exact timing of static constructor execution is implementation-dependent, but is subject to the following rules:
	The static constructor for a class executes before any instance of the class is created.
	The static constructor for a class executes before any of the static members for the class are referenced.
	The static constructor for a class executes after the static fields initializes for the class.
	The static constructor for a class executes at most one time during a single program execution.
	Example:-
	class A{
	static A() // A static constructor is parameter_less and no access modifiers are
	                                    //are allowed with its declaration.
	  {
	    Console.WriteLine("static constructor A is called");
	  }
	  public static void Main() { }	

```
}
```

Copy Constructor

A copy constructor copies the data from one object to another. In C#, there is no built in support for copy constructor. So we must create a user defined copy constructor if you wish to add this feature to a class. The following example illustrate the use of a copy constructor:

```
class complex_number {
private int real, img;
public complex_number() {real=0; img=0; }
public complex_number(complex_number c) //copy constructor
{
real=c.real;
img=c.img;
}
public void disp(){System.Console.WriteLine(real);
System.Console.WriteLine(img); }
public static void Main(){
complex_number c1=new complex_number(2,3);
complex_number c2=new complex_number(c1);
c2.disp(); } }
```

The above constructor is invoked by instantiating an object of type complex_number and passing it the object to be copied.

d. Explain one dimensional and two dimensional arrays with proper syntax and example.

Creating a one-dimensional array.

A one-dimensional array is used to store a list of values. The general syntax for declaring a one-dimensional array is:

datatype[ ] arrayname=new datatype[number of elements];

Example:-

```
int[] x=new int[5];
```

Initializing one-dimensional arrays

Syntax:-

arrayname[index]=value;

Example:-

```
int[] x=new int[3];     x[0]=100;   x[1]=23;    x[2]=440;
```

An array can be initialized at the time of its declaration in the following manner.

```
int[] x={100,23,440};
          or
int[] x=new int[3]{100,23,440};
          or
int[] x=new int[]{100,23,440};
```

Creating a two-dimensional array.

The two-dimensional arrays are used to store table of values. The general syntax for declaring a two-dimensional array is:

datatype[,] arrayname=new datatype[row size, column size];

Eg:- int[,] x=new int[5,4];

The above statement creates an array of 20 cells (Similar to a table of five rows and four columns).

Initializing two-dimensional arrays

Syntax:- **arrayname[rowindex, colindex]=value;**

Eg:- int[,] x=new int[3, 2]; x[0,0]=100; x[0,1]=23;

The initialization of a two-dimensional array at the time of declaration is done in the following manner.

```
int[,] x={{1,2},{2,4},{5,6}};      or       int[,] x=new int[2,3]{ {1,2},{2,4},{5,6}};
```

e. Define inheritance. Explain single inheritance with example.

f. List various reference types in c#. Also explain boxing operation with example.

<u>List of reference type</u>
interface type
delegate type
array type
class type
strings
The reference type hold references to the values stored somewhere in memory. You can imagine a reference type similar to a pointer that is implicitly dereferenced when you access them.

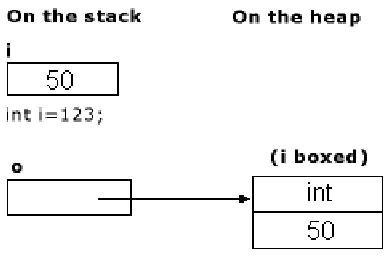
Boxing is an implicit conversion of a value type on the stack to an object on the heap. Boxing a value allocates an object instance and copies the value into the new object. Boxing enables value type variables to act like objects. Consider the following declaration of a value-type variable:

```
int i = 123;
```

The following statement implicitly applies the boxing operation on the variable i:

object o = i;

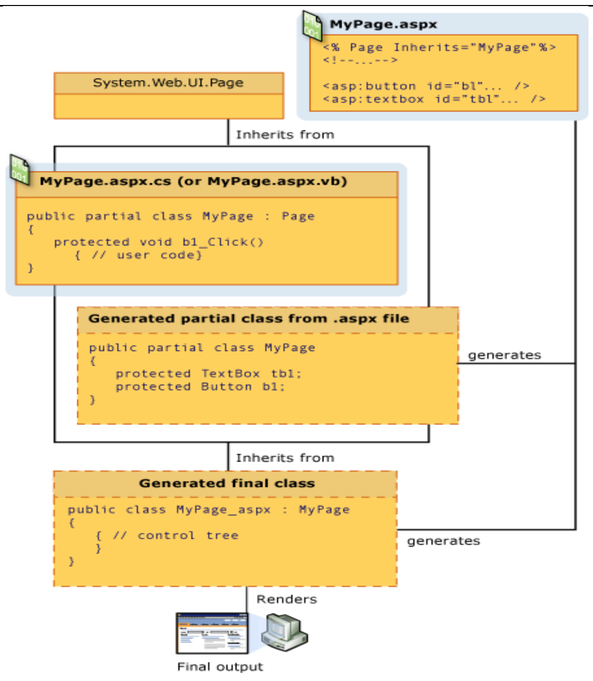
The result of this statement is creating an object o, on the stack, that references a value of the type int, on the heap. This value is a copy of the value-type value assigned to the variable i.



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

a. What are .aspx files? (1 Marks) Explain code behind class file in detail.(4 Marks)

In ASP.NET web applications, the UI code is present in the .aspx file. This file mainly contains HTML and control markups. The program logic is written in a separate file called code behind file. The code behind file has the extension . aspx.cs. There is a code behind file associated with each page of a website. As the program logic is separated from user interface design code, this model is easy to understand. The code for the page is compiled into a separate class from which the .aspx file derives. Only server controls can interact with the code behind the page; not the HTML controls.



```
<% Page Inherits="MyPage"%>
<!--...-->

<asp:button id="bl"... />
<asp:textbox id="tbl"... />
```

```
public partial class MyPage : Page
{
   protected void b1_Click()
     { // user code}
}
```

```
public partial class MyPage
{
   protected TextBox tb1;
   protected Button b1;
}
```

```
public class MyPage_aspx : MyPage
{
   { // control tree
   }
}
```

b. Explain each of the following in brief:

i. Web forms
ii. Post back
iii. Page rendering
iv. Page Load event
v. Page PreRender event

The Web Forms are web pages built on the ASP.NET Technology. Web Forms are pages that users request using their browser. These pages can be written using a combination of HTML, client-script, server controls, and server code.

The process of submitting an ASP.NET page to the server for processing is termed as post back. The post back originates from the client side browser. The Page object has an **IsPostBack** property, which is tells whether the request is a fresh request or a post back.

The user sends a request for a page to the server through the browser. The server inspects the request and compiles it. The response object is created and renders the HTTP response back to the browser. The browser receives the response, and parses the HTML in the response.

Page_Load

- Load is raised for page first and recursively for all child controls.
- In the event handler associated with this event you can write code that needs to be executed for each postback of the web page.

protected void Page_Load(object sender, EventArgs e){}

PreRender

- Raised after the Page object has created all controls that are required in order to render the page.
- It occurs for each control on the page.
- It is raised for the Page first and recursively for all child controls.
- This event can be used to make final changes to the contents of the page or its controls before rendering the page.
- This event can be handled by Page_PreRender handler.

protected void Page_PreRender(object sender, EventArgs e){}

c. List any four category of server controls in asp.net? Explain common properties of web server controls.

<u>Category of server controls</u>
Standard controls
Data controls
Validation controls
Navigation controls

Property	**Description**
AccessKey	Enables you to set a key with which a control can be accessed at the client by pressing the associated letter.
BackColor, ForeColor	Enables you to change background color and text color of a control.
BorderColor	This property is used to change the border color of a control.
BorderStyle	Using this property border Style can be set to none, dotted, dashed, solid double, groove etc.
BorderWidth	Enables you to change border width of a control.
CssClass	Enables you to set the style sheet class for this control.
Enabled	Determines whether the control is enabled or not. If the control is disabled user cannot interact with it.
Font	Enables you to change the Font settings.
Height	Enables you to set the height of the control.
Width	Enables you to set the width of the control.
ToolTip	This property enables you to set a tooltip for the control in the browser and is rendered as a title attribute in the HTML, is shown when the user hovers the mouse over the control.
Visible	Determines whether the control is visible or not.

d. Explain the basic functionality of each of the following webserver controls.
i. CheckBox
ii. TextBox
iii. DropDownList
iv. Menu

e. Write a brief note on various validator controls.

They are web server controls used for the validation of form fields. These controls are included in **System.Web.UI.WebControls** namespace. They can perform both server-side and client-side validation.

<u>RequiredFieldValidator</u>
- It ensures that the user has entered a value in the associated input control.
- Use the RequiredFieldValidator control to make an input control a mandatory field.
- By default, the initial value is an empty string (""), which indicates that a value must be entered in the input control for it to pass validation.

<u>RangeValidator</u>
- The RangeValidator control is used to check whether the user enters an input value that is within a specified range.
- Applicable for range of numbers, dates, and characters.

<u>CompareValidator</u>
- The CompareValidator control is used to compare the value of one input control to the value of another input control or to a fixed value.

<u>CustomValidator</u>
- Validation is done based on a validation function.

- The control allows you to validate both client side and server side.

RegularExpressionValidator

- The RegularExpression Validator control is used to ensure that an input value matches a specified pattern.
- These patterns are called regular expressions or regex.
- Patterns consists of literal characters and special characters.

Literal characters are normal characters with no special meaning whereas special characters have special meaning in the pattern.

f. What are rich controls? Explain about Calendar and AdRotator controls in brief.

Rich controls are web controls that model complex user interface elements. A typical rich control can be programmed as a single object but renders itself using a complex sequence of html elements. Rich controls can also react to user actions and raise more-meaningful events that your code can respond to on the web server. In other words, rich controls give you a way to create advanced user interfaces in your web pages without writing lines of convoluted html.

The Calendar control presents a miniature calendar that you can place in any web page. Like most rich controls, the Calendar can be programmed as a single object, but it renders itself with dozens of lines of HTML output.

```
<asp:Calendar id="MyCalendar" runat="server" />
```

The Calendar control presents a single-month view. The user can navigate from month to month by using the navigational arrows, at which point the page is posted back and ASP.NET automatically provides a new page with the correct month values.

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.

3. Attempt _any three_ of the following: **15**

a. Explain exception handling mechanism with proper syntax and example.

An exception is an event that interrupts normal program execution. There are two aspects involved in the processing of exceptions: raising (or throwing) the exception and handling the exception. The steps involved in exception handling are:
Detect problem-causing exception (hit the exception).
Inform that an error has occurred (throw the exception).
Receive error information (catch the exception).
Take corrective actions (handle the exception).
In C#, the try and catch block used for handling exception has the following syntax:

Try with multiple catch block	Try block with catch and finally block	Try block with only finally block

try{ statements; }
catch(ExceptionType obj)
{ statements; }
catch(ExceptionType obj)
{ statements; }
	try{ statements; }
	catch(ExceptionType obj)
	{ statements; }
	Finally { statements; }
		try
		{ statements; }
		finally {
		statements; }

A try block must have a catch block or a finally block. There can be multiple catch blocks but a single finally block for a try block. The normal scope rules apply for the try block. For instance any local variable defined in the try block will not be accessible outside the block.

Example :-

```
try{
int n=Convert.ToInt32(Console.ReadLine());
System.Console.WriteLine("n="+n);
}catch(FormatException e) { }
```

b. Describe session state variables in asp.net.

Each HTTP request for a page is treated as an independent request as HTTP is a stateless protocol. The server retains no knowledge of variable values that were used during previous requests. Session state identifies requests from the same browser during a limited time window as a session, and provides a way to persist variable values for the duration of that session. It is a server-side state management technique.

Adding session state variables:

```
Session["session_variable_name"]=value;
```

Example:-

```
Session["x"] ="Tata";
Session["y"] = "atat";
```

- The session variable can also be added using add() method.

  Example:-

  ```
  string s1 = TextBox1.Text;
  Session.Add("z", s1);
  ```

Retrieving Session Variables

- String variable=Session["session_variable_name"].ToString();

  Example:-

  ```
  string x = Session["z"].ToString();
  ```

c. Explain state management through persistent cookies.

A cookie is a technique used for client side state management. It helps Web applications to store user-specific information. It is a text file stored on the client side by the browser. These files store name value pairs. A Web application can read the information stored on the cookie whenever the user visits the site. Cookies are of two types

- Temporary (Transient or session) cookies

  It is alive in the main memory as long as the user’s session is alive.
- Permanent cookies

  It is stored on the client disc and its expiration date determines how long the cookie remains on the client’s computer.

Creating and Reading Cookies using Request and Response Objects

Cookies property of Response and Request objects are used to manipulate HttpCookieCollection object.

Syntax for Creating a Cookie:

```
Response.Cookies[“cookie name”].Value=“specify value”;
```

Example:-

```
Response.Cookies["uname"].Value = TextBox1.Text;
```

Syntax for Reading a Cookie:

```
String str=Request.Cookies[“cookie name”].Vaue;
```

Example:-

```
string str= Request.Cookies["pwd"].Value;
```

d. Describe the features and benefits of master pages in asp.net.

Master page provides a framework (common content as well as the layout) 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 consist 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.

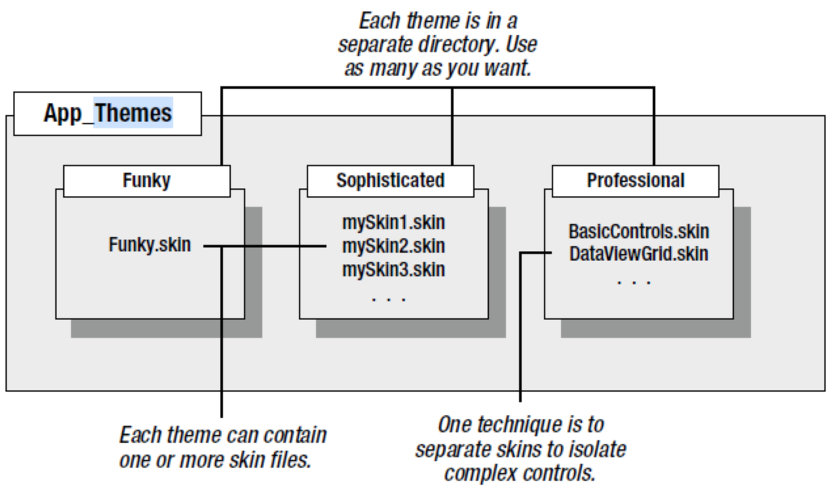
Advantages of Master Page

- They allow you to centralize the common functionality of your pages so that you can make updates in just one place.
- They make it easy to create one set of controls and code and apply the results to a set of pages. For example, you can use controls on the master page to create a menu that applies to all pages.
- They give you fine-grained control over the layout of the final page by allowing you to control how the placeholder controls are rendered.
- They provide an object model that allows you to customize the master page from individual content pages.

e. What is a theme? How does it work?

Like CSS, themes allow you to define a set of style details that you can apply to controls in multiple pages. However, unlike CSS, themes aren't implemented by the browser. Instead, ASP.NET processes your themes when it creates the page. All themes are application specific. To use a theme in a web application, you need to create a folder that defines it. This folder needs to be placed in the App_Themes folder, which must be placed inside the top-level directory for your web application. An application can contain definitions for multiple themes as long as each theme is in a separate folder. Only one theme can be active on a given page at a time. To actually make your theme accomplish anything, you need to create at least one skin file in the theme folder. A *skin file* is a text file with the .skin extension. ASP.NET never serves skin files directly; instead, they're used behind the scenes to define a theme.

When you add a control tag for the ListBox in the skin file, it might look like following markup:
<asp:ListBox runat = "server" ForeColor = "White" BackColor = "Orange"/>
The runat = "server" portion is always required. Everything else is optional. You should avoid setting the ID attribute in your skin file because the page that contains the ListBox needs to define a unique name for the control in the actual web page.


Each theme is in a separate directory. Use as many as you want.
App_Themes
Funky
Funky.skin
Sophisticated
mySkin1.skin
mySkin2.skin
mySkin3.skin
. . .
Professional
BasicControls.skin
DataViewGrid.skin
. . .
Each theme can contain one or more skin files.
One technique is to separate skins to isolate complex controls.

f. What are the three different ways to use styles on web pages? Explain various category of style setting in New Style dialog box.

Web pages can use styles in three different ways:

Inline style: An inline style is a style that’s placed directly inside an HTML tag. This can get messy, but it’s a reasonable approach for one-time formatting. You can remove the style and put it in a style sheet later.
Internal style sheet: An internal style sheet is a collection of styles that are placed in the <head> section of your web page markup. You can then use the styles from this style sheet to format the web controls on that page. By using an internal style sheet, you get a clear separation between formatting (your styles) and content (the rest of your HTMLmarkup). You can also reuse the same style for multiple elements.
External style sheet: An external style sheet is similar to an internal style sheet, except it’s placed in a completely separate file. This is the most powerful approach because it gives you a way to apply the same style rules to many pages.

Category	Description
Font	Allows you to do all font settings
Block	Allows you to fine-tune additional text settings. Example:-text alignment, test indentation, spacing between letters and words, etc.
Background	Allows you to set a background color or image.
Border	Allows you to do border setting.
Box	Allows you to define the margin and the padding
Position	Allows you to set a fixed width and height for your element, and use absolute positioning to place your element at a specific position on the page.
Layout	Allows you to control miscellaneous layout settings.
List	If you’re configuring a list (a <ul> or <ol> element), you can set the numbering or bullet style. These settings aren’t commonly used in ASP.NET web pages because you’re more likely to use ASP.NET list controls like the BulletedList.
Table	Allows you to set details that only apply to table elements (such as <tr> and <td>). For example, you can control whether borders appear around an empty cell.

4. Attempt *<u>any two</u>* of the following: **10**

a. What is data binding? Explain repeated value data binding with example.

Data binding is the process how the user interface controls of a client application are configured to retrieve from, or update data into, a data source. It provides a simple and consistent way for applications to present and interact with data. If the user modifies the data in a data-bound control, your program can update the corresponding record in the database.
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 (such as a field in a data table). Then you call DataBind(), the control automatically creates a full list by using all the corresponding values.
Following are the steps.

1. Create and fill some kind of data object such as arrays, ArrayList, Hashtable,DataTable and DataSet
2. Link the object to the appropriate control.
3. Activate the binding using DataBind() method.

<u>A Simple List-Binding Example</u>

```
public List<string> GetCarsList()
{
```

```
List<string> cars = new List<string>();
cars.Add("Santro");  cars.Add("Marazzo"); cars.Add("i20");  cars.Add("Ertiga");
cars.Add("Brezza");  cars.Add("Tigor"); cars.Add("Creta");
return cars;
}
protected void Page_Load(object sender, EventArgs e)
{
lstItems.DataSource=GetCarsList();
this.DataBind();
}
```

b. Write any three similarities between FormView and DetailsView. Explain about item templates of FormView.

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.

ItemTemplate is used to display the data from data the source in ReadOnly mode. The controls included in an ItemTemplate are controls that only display data, such as a Label control. The template can also contain command buttons to change the FormView control’s mode to insert or edit, or to delete the current record. The ItemTemplate template can include Link Button controls that change the mode of the FormView control based on the Command Name value. A CommandName value of New puts the record into insert mode and loads the InsertItemTemplate, which allows the user to enter values for a new record. You can add a button with a CommandName value of Edit to the ItemTemplate template to put the FormView control into edit mode. A button with a CommandName value of Delete can be added to the ItemTemplate template to allow users to delete the current record from the data source.

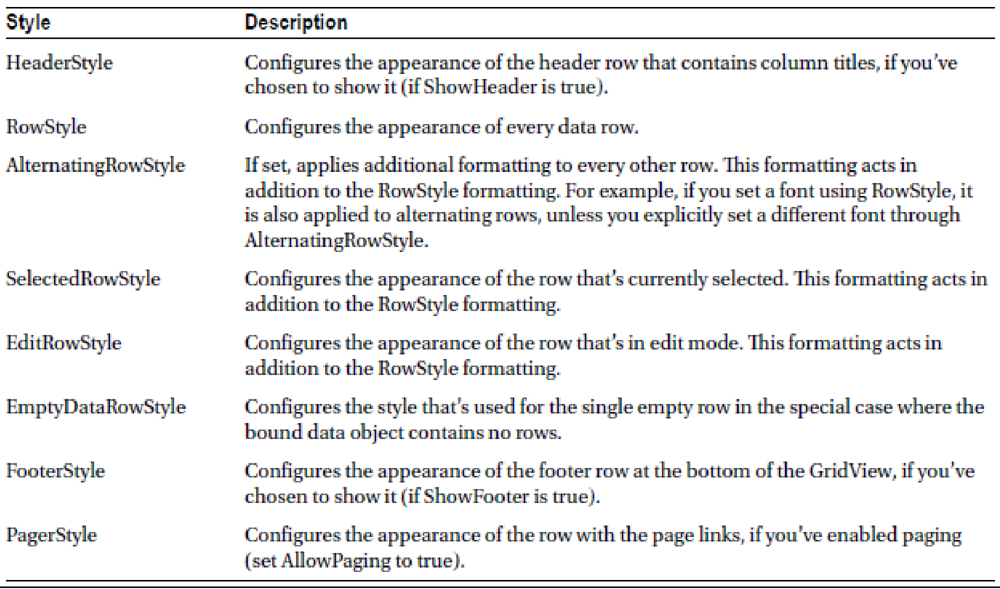
c. Explain data binding to a Dictionary collection.

- Dictionary class is present in System.Collection.
- A Dictionary collection uses key and value pairs of data.
- The values from Dictionary can be retrieved very easily using keys.
- In ordinary collections, such as the List, you need to find the item you want by its index number position.
- Also you may have to traverse through the whole collection until you come across the right item.
- But in dictionary collection an item can be retrieved by using its key.
- Ordinary collection is ideal when you want to work with all the items of the collection at once.

Dictionary collection is ideal when you want to work with a specific item.

```
protected void Page_Load(object sender, EventArgs e)
{
Dictionary<string,string> chatabb = new Dictionary<string,string>();
chatabb.Add("PM", "Private Message"); chatabb.Add("OMG", "Oh My God");
chatabb.Add("NYOB", "None of Your Business"); chatabb.Add("KIT", "Keep In
Touch");
chatabb.Add("TTYL", "Talk To You Later");
chatabb.Add("IDK", "I Don't Know");
ListBox1.DataSource = chatabb;
ListBox1.DataTextField = "Key";
this.DataBind();
}
```

d. Explain various style properties of GridView control.



Style	Description
HeaderStyle	Configures the appearance of the header row that contains column titles, if you've chosen to show it (if ShowHeader is true).
RowStyle	Configures the appearance of every data row.
AlternatingRowStyle	If set, applies additional formatting to every other row. This formatting acts in addition to the RowStyle formatting. For example, if you set a font using RowStyle, it is also applied to alternating rows, unless you explicitly set a different font through AlternatingRowStyle.
SelectedRowStyle	Configures the appearance of the row that's currently selected. This formatting acts in addition to the RowStyle formatting.
EditRowStyle	Configures the appearance of the row that's in edit mode. This formatting acts in addition to the RowStyle formatting.
EmptyDataRowStyle	Configures the style that's used for the single empty row in the special case where the bound data object contains no rows.
FooterStyle	Configures the appearance of the footer row at the bottom of the GridView, if you've chosen to show it (if ShowFooter is true).
PagerStyle	Configures the appearance of the row with the page links, if you've enabled paging (set AllowPaging to true).

e. Briefly explain following features of GridView control.

i. Sorting
ii. Paging
iii. Selecting a row
iv. Editing with the GridView

1) Sorting in GridView
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.

2) Paging in GridView
Paging refers to the ability of GridView control to display bound data one page at a time.The 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 you want in each page.

3) Selecting a GridView Row
Selecting an item refers to the ability to click a row and become highlighted to indicate that the user is currently working with this record by setting AutoGenerateSelectButton attribute of GridView to True.
4) Editing with the GridView
Set AutoGenerateEditButton property of GridView to true to get Edit button on each row. Also to allow updation of data in the actual data base through a data-bound GridView, an UpdateCommand should be specified in the SqlDataSource as follows:
UpdateCommand="update CollegeList set CollegeName=@CollegeName, City=@City where CollegeCode=@CollegeCode"

f. Describe asp.net provider model and direct data access method.

ADO.NET has a set of classes that are used to connect to a specific data source. Each set of data interaction classes is called an ADO.NET data provider. ADO.NET data provider includes SQL Server Data Provider, Oracle Data Provider, OLEDB Data Provider and ODBC Data Provider. The classes used to connect to a specific data source includes Connection, Command, DataReader and DataAdapter.

- Connection is used to establish a connection to a specific data source.
- Command is used to execute SQL statements against the data source.

- DataReader is used to read data from data source.
- DataAdapter populates a DataSet and resolves updates with the data source.

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.

Using Direct Data Access

To query information with simple data access, follow these steps:

1. Create Connection, Command, and DataReader objects.
2. Use the DataReader to retrieve information from the database, and display it in a control on a web form.
3. Close your connection.
4. Send the page to the user. At this point, the information your user sees and the information in the database no longer have any connection, and all the ADO.NET objects have been destroyed.

To add or update information, follow these steps:

1. Create new Connection and Command objects.
2. Execute the Command (with the appropriate SQL statement).

5.	**Attempt *any two* of the following:**	**10**
a.	Briefly explain different types of authentication in asp.net.	

b. What is XML? What are its basic characteristics?

- XML elements are composed of a start tag and an end tag. Content is placed between the start and end tags.

  Eample:-

  <ProductName> Camera **</ProductName>** //element composed of start tag and an end tag

  <ProductName /> //combined start tag and an end tag representing empty element
- Whitespace between elements is ignored.
- You can use only valid characters in the content for an element. (You can't enter special characters, such as the angle brackets (< >) and the ampersand (&), as content.)

  use &lt for <, &gt for >, &amp for &
- XML elements are case sensitive.
- All elements must be nested in a root element.
- Every element must be fully enclosed.
- XML documents usually start with an XML declaration like <?xml version="1.0"?>. This signals that the document contains XML and indicates any special text encoding. However, many XML parsers work fine even if this detail is omitted.

c. What is the use of XMLTextWriter class? Explain various methods of this class.

The XmlTextWriter class allows you to write to XML files. The Constructor for creating an instance of the XmlTextWriter class is:

XmlTextWriter (Stream, Encoding)

If encoding is null it writes out the stream as UTF-8.

```
string file =Path.Combine(Request.PhysicalApplicationPath,"XMLFile.xml");
FileStream fs = new FileStream(file, FileMode.Create);
XmlTextWriter w = new XmlTextWriter(fs, null);
```

WriteStartDocument()	Writes the XML declaration with the version "1.0"
WriteEndDocument()	Closes any open elements or attributes and puts the writer back in the Start state.
WriteStartElement(String)	Writes out a start tag with the specified local name.
WriteEndElement()	Closes an element
WriteString(String)	Writes the given text content.
WriteAttributeString(String, String)	Writes out the attribute with the specified local name and value.
WriteComment(String)	Writes out a comment <!--...--> containing the specified text.

d. What is authorization? Explain adding authorization rules in web.config file.

Authorization: Authorization is the process of determining whether a user has sufficient permission to perform a given action. To control who can and can’t access your website, you need to add access-control rules to the <authorization> section of your web.config file.

```
<configuration>
. . .
<system.web>
. . .
<authentication mode="Forms">
<forms loginUrl="~/Login.aspx" />
</authentication>
<authorization>
<allow users="*" />
</authorization>
</system.web>
</configuration>
```

The asterisk (*) is a wildcard character that explicitly permits all users to use the application—even those who haven’t been authenticated. The question mark (?) is a wildcard character that matches all anonymous users. By including following rule in applications web.config file, you specify that anonymous users are not allowed.

```
<authorization>
<deny users="?" />
</authorization>
```

One can add more than one rule to the authorization section:

```
<authorization>
<allow users="*" />
<deny users="?" />
</authorization>
```

e. Explain the working of update progress control in detail.

The UpdatePanel performs its work asynchronously in the background. While the asynchronous request is under way, the user won’t necessarily realize that anything is happening. If the asynchronous request takes some time, the user may assume the page is broken or click the same button multiple times.This creates needless extra work for your web application and slowing down the process further. The UpdateProgress control can be used to overcome this problem. The UpdateProgress control allows you to show a message while a time-consuming update is under way. It provides a wait message showing that the last request is still being processed.
Example:

```
<form id="form1" runat="server">     <div>
```

```
   <asp:ScriptManager ID="ScriptManager1" runat="server">
   </asp:ScriptManager>  <br />  <br />
   <asp:UpdatePanel ID = "UpdatePanel1" runat = "server">
         <ContentTemplate>
                  <asp:Label ID = "Label1" runat = "server" Font-Bold = "True" >
                  </asp:Label> <br /><br />
                  <asp:Button ID = "Button1" runat = "server" Text = "Click Here to Start"
                   onclick="Button1_Click" />
         </ContentTemplate>
    </asp:UpdatePanel><br />
   <asp:UpdateProgress runat = "server">
          <ProgressTemplate>
          <b>Loading please wait …</b>
          </ProgressTemplate>
   </asp:UpdateProgress>
   </div>
   </form>
```

f. Explain about implementation of timed refreshes of update panel using timer.

Timer control can be used to refresh an UpdatePanel at regular intervals without waiting for a user action. Interval property of Timer can be set as follows:

```
<asp:Timer ID = "Timer1" runat = "server" Interval = "1000" />
```

To use the timer with partial rendering, wrap the updateable portions of the page in UpdatePanel controls with the UpdateMode property set to Conditional. Then add a trigger that forces the UpdatePanel to update itself whenever the Tick event of Timer occurs.

Following is the markup for the same:

```
<asp:UpdatePanel ID = "UpdatePanel1" runat = "server" UpdateMode =
"Conditional">
<ContentTemplate>
...
</ContentTemplate>
<Triggers>
<asp:AsyncPostBackTrigger ControlID = "Timer1" EventName = "Tick" />
</Triggers>
</asp:UpdatePanel>
```