1A) Which of the following can be indicated in the Page Directive of the .aspx file in a web page

a) Where the design file is located. b) The way in which event handlers are connected. c) The name of your custom page class. d) Version of .NET framework. e) Value for PreviousPageType in case of information transfer.	i. b, c & e only **ii. b & c only** iii. a, b & e only iv.a, b & c only

1B) With respect to _doPostBack( ) function which are not true from below list ?

a) This function has the responsibility of setting the values of EVENTTARGET and EVENTARGUMENT fields with appropriate information about the event. b) This function is added by ASP.NET to every web page that includes a Button web control since AutoPostBack property is by default True for a Button. c) ASP.NET automatically changes a client-side Javascript event into a server-side ASP.NET event using the _doPostBack( ) function as intermediary. d) Any control that has its AutoPostBack property set to true is connected to the _doPostBack( ) function using the onclick attribute only. e) _doPostBack( ) is a client-side Javascript function added in the .aspx file of the web form.	i. a & c only ii. b, c & e only iii. b & e only **iv. b, d & e only**

1C) Which of these is not a possible value for the DisplayMode property of a Validation Summary control?

i.Single Paragraph   ii.List   **iii.Numbered List**   iv.BulletList

1D) To get "Enter the number <here>" displayed on the browser using a Paragraph tag with id="Result" which of these can to be used alternatively in the .cs file?

a) Result.InnerHtml = Server.HtmlEncode("Enter a number <here>");
b) Result.InnerText = Server.HtmlEncode("Enter a number <here>");
c) Result.InnerHtml = "Enter a number <here>";
d) Result.InnerText = "Enter a number <here>";
	i. b & c
	ii. a & d
	iii. b & a
	iv. c & d

1E) For the below mentioned code snippet (performing multiplication), which is true:

Note: CE-Compilation Error   LE-Logical Error

```
class Program {
        public void MulNum (int number, out int res) {
        res = number * res;  }
  static void Main (string[] args){
        Program p1 = new Program();
        Console.WriteLine("Enter a number");  string inp=Console.ReadLine();
        int num, result;   int.TryParse(inp, out num);
        p1.MulNum(num, out result);
        Console.WriteLine("The result of multiplication is {0}", result);  Console.Read(); } }
```

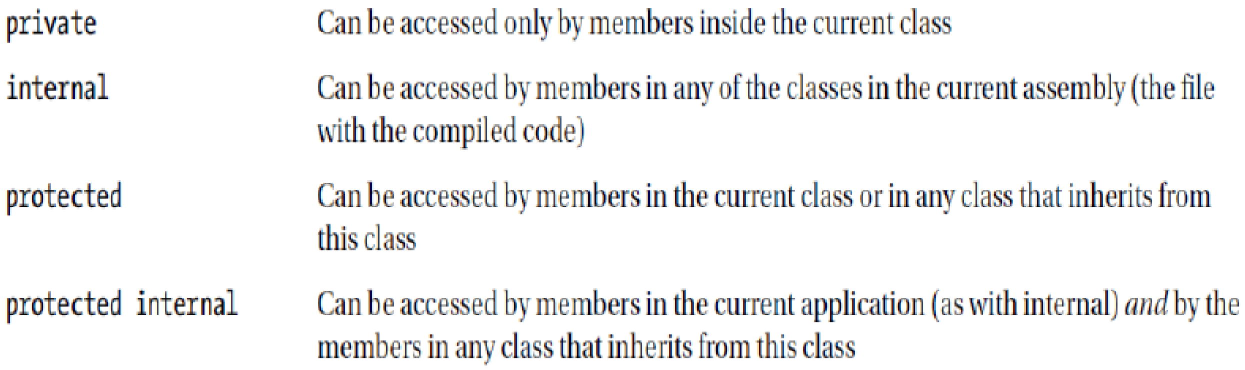
i.No CE & no LE

ii. No CE & only LE

iii. CE & no LE

iv. CE & LE

2A. **(0.5 Mark each)**



private	Can be accessed only by members inside the current class
internal	Can be accessed by members in any of the classes in the current assembly (the file with the compiled code)
protected	Can be accessed by members in the current class or in any class that inherits from this class
protected internal	Can be accessed by members in the current application (as with internal) *and* by the members in any class that inherits from this class

2B.

```
class Program
{
    static void Main(string[] args)
    {
         List<Student> student = new List<Student>();           (0.5 M)
        string rno,nm;
         int x;
        for (int i = 0; i < 4; i++)
        {
            Console.WriteLine("Enter RollNo");
            rno = Console.ReadLine();
            Console.WriteLine("Enter Name");
            nm = Console.ReadLine();
            Student stud = new Student();
            stud.RollNo = rno;
            stud.Name = nm;

            if (int.TryParse(stud.RollNo,out x))                (0.5 M)
                  student.Add(stud);

        }

        foreach (Student  st in student)                        (0.5 M)
        {
            Console.WriteLine(st.RollNo);
        }

        Console.ReadLine();
    }
}
class Student                                                   (0.5 M)
{
  private string rollno;

  public string RollNo {                                   (0.5 M)
     get
     { return rollno ;  }
    set
     {
        if (Convert.ToInt32(value.Substring(0, 2)) >= 10)
           rollno = value;
        else
           Console.WriteLine("RollNo is expected");
     } }
  public string Name { get; set; }                         (0.5 M)
   }}
```

3A. **Response.Redirect("www.google.com");** **(0.5 M)**

When you use the Redirect() method, ASP.NET immediately stops processing the page and sends a redirect message back to the browser. When the browser receives the redirect message, it sends a request for the new page.

Server.Transfer("newpage.aspx"); **(0.5 M)**

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.

Web.config: (Syntax - 0.5 M) (any two point - 0.5 M)

- The web.config file uses a predefined XML format.
- This file is case-sensitive
- You can update web.config settings at any point, even while your application is running.
- The entire content of the file is nested in a root <configuration> element.

```
  <appSettings > . . .</appSettings>
<connectionStrings > . . .</connectionStrings>
<system.web > . . .</system.web>
```

3B. **<u>.cs file</u>**

```
 public partial class sess1 : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
   {
      Table1.Visible = false;
      DropDownlist1.Items.Add("2");                0.5M
      DropDownlist1.Items.Add("3");
      DropDownlist1.Items.Add("4");
   }
   protected void Button1_Click(object sender, EventArgs e)
   {
      Table1.Visible = true;
      int a = int.Parse ( TextBox1.Text);          0.5M
      int b = int.Parse(TextBox2.Text);

      Table1.Height = Unit.Pixel(a);
      Table1.Width  = Unit.Pixel(b);                  0.5M

      int rows = int.Parse(DropDownList1.SelectedItem.Text);  0.5M
      for (int i = 0; i < rows; i++)
      {
         TableRow r1 = new TableRow();
         Table1.Controls.Add(r1);                         0.5M
         TableCell data = new TableCell();

         data.BorderStyle = BorderStyle.Inset;
         data.Text = "Row " + (i+1);                          0.5M
         r1.Controls.Add(data);
      }}}
```

4A. **(0.5 X 4 =2) // each point 0.5 mark**

Session state can be lost in several ways:

- If the user closes and restarts the browser.
- If the user accesses the same page through a different browser window, although the session will still exist if a web page is accessed through the original browser window. Browsers differ on how they handle this situation.
- If the session times out due to inactivity. More information about session timeout can be found in the configuration section.
- If your web page code ends the session by calling the Session.Abandon() method.

4B.

RequiredFieldValidator -0.5 , RegularExpressionValidator – 0.5, CompareValidator -0.5 , .cs file – 0.5, ShowMessageBox -0.5, ShowSummary -0.5

.aspx

```
<asp:Label ID="lpass" runat="server" Text="Password"></asp:Label>
<asp:TextBox ID="txt_pass" runat="server" ></asp:TextBox>
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" Display="None" runat="server" ErrorMessage="Please enter a password" ControlToValidate="txt_pass"></asp:RequiredFieldValidator>
<asp:RegularExpressionValidator Display="None" ID="RegularExpressionValidator1" runat="server" ErrorMessage="Password can have 6-10 characters only" ValidationExpression="\w{6,10}" ControlToValidate="txt_pass"></asp:RegularExpressionValidator>
<br /><br />
<asp:Label ID="lconfirm" runat="server" Text="Confirm Password :"></asp:Label>
<asp:TextBox ID="txt_confirm" runat="server" ></asp:TextBox>
```

```
        <asp:RequiredFieldValidator ID="RequiredFieldValidator2" Display="None"
runat="server" ErrorMessage="Please enter the password for confirmation"
ControlToValidate="txt_confirm"></asp:RequiredFieldValidator>
        <asp:CompareValidator ID="CompareValidator1" Display="None" runat="server"
ErrorMessage="PleaseEnterthe correct password for confirmation"
ControlToValidate="txt_confirm"
ControlToCompare="txt_pass"></asp:CompareValidator>
        <br /><br />
        <asp:Button ID="btn_submit" runat="server" Text="Submit" OnClick="btn_submit_click"
/>
        &nbsp;
        <asp:Button ID="btn_clear" runat="server" Text="Clear" OnClick="btn_clear_click" />
        <br />
        <asp:Label ID="lmsg" runat="server"></asp:Label>
        <asp:ValidationSummary ID="ValidationSummary1" runat="server"
ShowMessageBox="true" ShowSummary="false" />
```

.cs file

```
protected void btn_submit_click(object sender, EventArgs e)
    {
        if (Page.IsValid)
        {
            lmsg.Text = "Successfully validated !";
        }
    }
    protected void btn_clear_click(object sender, EventArgs e)
    {
        txt_pass.Text = "";
        txt_confirm.Text = "";
    }
```