CMR
INSTITUTE OF
TECHNOLOGY

USN	1	C								

Internal Assessment Test 3 – December 2020

Sub:	Programming using C#.NET							Code:	18MCA51
Date:	18-12-20	Duration:	90 mins	Max Marks:	50	Sem:	V A & B	Branch:	MCA

Note: Answer any 5 questions. All questions carry equal marks. Total marks: 50

		Marks	OBE CO	OBE RBT
1.	a Discuss panel control with respect to windows forms.	6	CO3	L3

The Panel Control is a container or grouping mechanism in Windows Forms, and can be used both for visual and logical organization

Windows Forms Panel controls are used to provide an identifiable grouping for other controls. Typically, you use panels to subdivide a form by function. The Panel control is similar to the GroupBox control; however, only the Panel control can have scroll bars, and only the GroupBox control displays a caption. The Panel Control is a container control to host a group of similar child controls. One of the major uses I have found for a Panel Control is when you need to show and hide a group of controls. Instead of show and hide individual controls, you can simply hide and show a single Panel and all child controls.

Windows Forms Panel controls are used to provide an identifiable grouping for other controls. Typically, you use panels to subdivide a form by function. For example, you may have an order form that specifies mailing options such as which overnight carrier to use. Grouping all options in a panel gives the user a logical visual cue. At design time all the controls can be moved easily — when you move the Panel control, all its contained controls move, too. The controls grouped in a panel can be accessed through its Controls property. This property returns a collection of Control instances, so you will typically need to cast a control retrieved this way to its specific type.

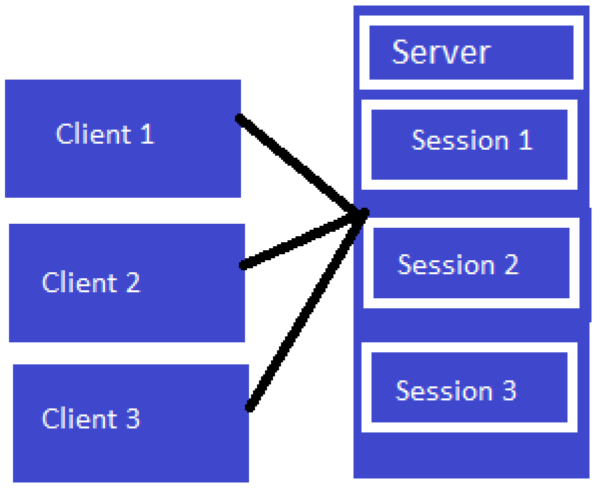
This control is very useful because it works like container of every control. Assume we have very long content page and wishing to display it in very short area on web page, surely we can use it by enabling ScrollBars property.

Group boxes (class GroupBox) and panels (class Panel) arrange components on a GUI. For example, buttons related to a particular task can be placed inside a group box or panel. All these buttons move together when the group box or panel is moved. The main difference between the two classes is that group boxes can display a caption, and panels can have scrollbars. The scrollbars allow the user to view additional controls inside the panel by scrolling the visible area. Group boxes have thin borders by default, but panels can be set to have borders by changing their BorderStyle property.
To create a panel, drag it onto the form and add components to it. To enable the scroll-bars, set the Panel’s AutoScroll property to true in the Properties window. If the panel is resized and cannot hold its controls, scrollbars appear. These scroll-bars then can be used to view all the components in the panel (both when running and designing the form). This allows the programmer to see the GUI exactly as it appears to the client.

		Marks	CO	RBT
	b Write a short note on sessions in ASP.NET.	4	CO3	L3

Session is a State Management Technique. A Session can store the value on the Server. It can support any type of object to be stored along with our own custom objects. A session is one of the best techniques for State Management because it stores the data as client-based, in other words the data is stored for every user separately and the data is secured also because it is on the server.

Sessions can be used to store even complex data for the user just like cookies. Sessions can be used easily in ASP.NET with the Session object. Session variables are stored in a SessionStateItemCollection object that is exposed through the HttpContext.Session property. In an ASP.NET page, the current session variables are exposed through the Session property of the Page object.
The collection of session variables is indexed by the name of the variable or by an integer index. Session variables are created by referring to the session variable by name. We do not have to declare a session variable or explicitly add it to the collection. The following example shows how to create session variables in an ASP.NET page for the first and last name of a user, and set them to values retrieved from TextBox controls.



```
Session["FirstName"] = FirstNameTextBox.Text;
Session["LastName"] = LastNameTextBox.Text;
```

Session variables can be any valid .NET Framework type. Sessions are identified by a unique identifier that can be read by using the SessionID property. When session state is enabled for an ASP.NET application, each request for a page in the application is examined for a SessionID value sent from the browser. If no SessionID value is supplied, ASP.NET starts a new session and the SessionID value for that session is sent to the browser with the response.
By default, SessionID values are stored in a cookie. However, we can also configure the application to store SessionID values in the URL for a "cookieless" session.
A session is considered active as long as requests continue to be made with the same SessionID value. If the time between requests for a particular session exceeds the specified time-out value in minutes, the session is considered expired. Requests made with an expired SessionID value result in a new session. We will re-use the cookie example, and use sessions instead. The sessions will expire after a certain amount of minutes, as configured in the web.config file. Markup code. And here is the CodeBehind:

```
publicpartialclass _Default : System.Web.UI.Page {
        protectedvoidPage_Load(object sender, EventArgs e)
      {
      if(Session["BackgroundColor"] != null)
      {
        ColorSelector.SelectedValue = Session["BackgroundColor"].ToString();
      BodyTag.Style["background-color"] = ColorSelector.SelectedValue;
```

```
        }
    }
protectedvoidColorSelector_IndexChanged(object sender, EventArgs e)
    {
    BodyTag.Style["background-color"] = ColorSelector.SelectedValue;
    Session["BackgroundColor"] = ColorSelector.SelectedValue;
    }
}
```

Please notice that session values are tied to an instance of your browser. If we close down the browser, the saved value(s) will usually be "lost". Also, if the webserver recycles the aspnet_wp.exe process, sessions are lost, since they are saved in memory as well. This can be avoided by saving session states on a separate StateServer or by saving to a SQL server.

Advantages of Session
It store session data in a memory object of the current application domain. So accessing data is very fast and data is easily available.
Implementation is very easy, similar to using the ViewState

2. a

Explain the implementation of combo box in C#. 10 CO3 L3

A ComboBox displays a text box combined with a ListBox, which enables the user to select items from the list or enter a new value. The user can type a value in the text field or click the button to display a drop down list. You can add individual objects with the Add method.

C# controls are located in the Toolbox of the development environment, and you use them to create objects on a form with a simple series of mouse clicks and dragging motions. A ComboBox displays a text box combined with a ListBox, which enables the user to select items from the list or enter a new value. The user can type a value in the text field or click the button to display a drop down list. You can add individual objects with the Add method. You can delete items with the Remove method or clear the entire list with the Clear method.

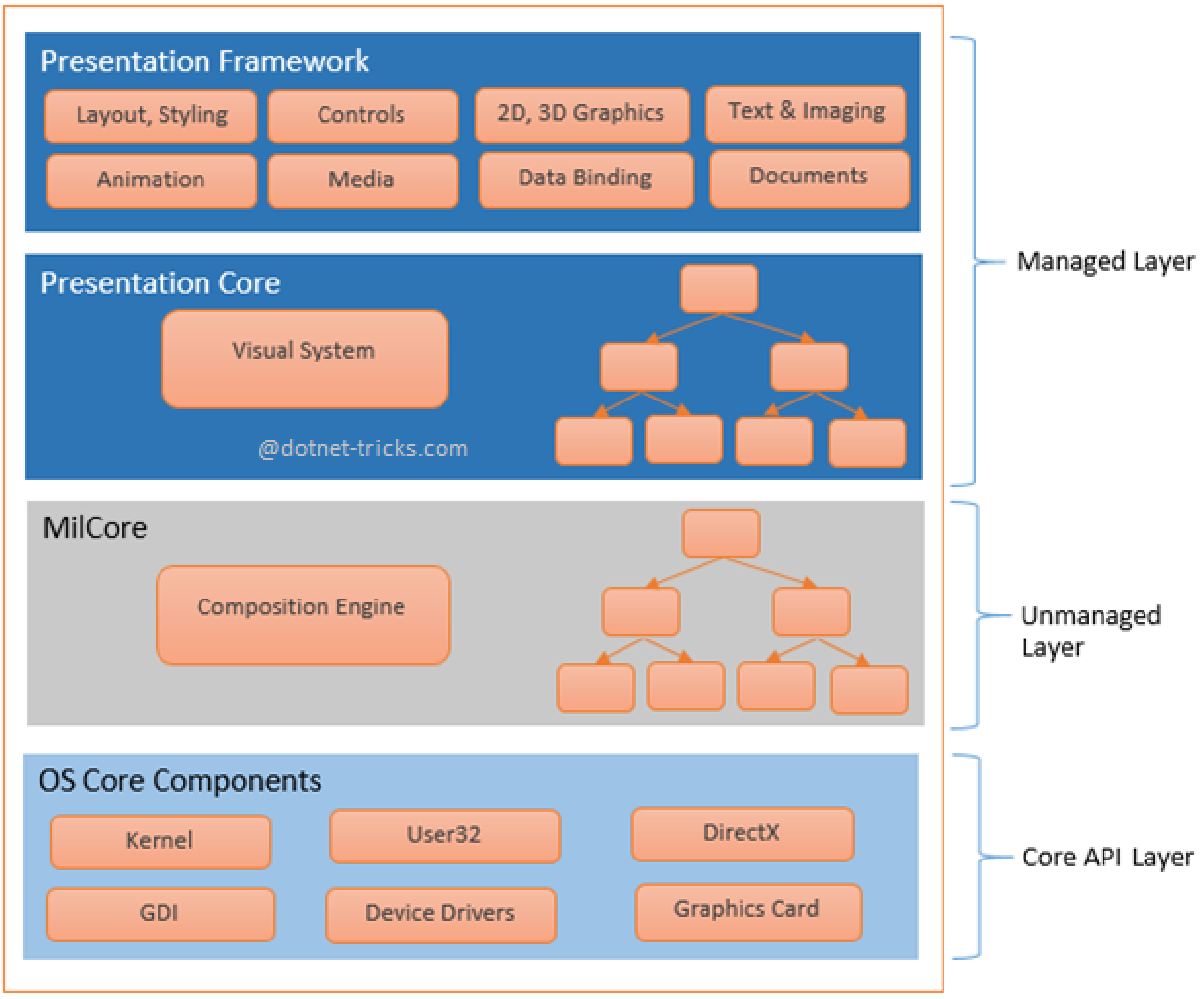
```
comboBox1.Items.Add("Sunday");
comboBox1.Items.Add("Monday");
comboBox1.Items.Add("Tuesday");
```

```
string var;
var = comboBox1.Text;
                    Or
var item = this.comboBox1.GetItemText(this.comboBox1.SelectedItem);
MessageBox.Show(item);
```

3.

a Explain architecture of WPF. 10 CO4 L4

Windows Presentation Framework is a next generation UI framework to create applications with a rich user experience. It is part of the .NET framework 3.0 and higher. WPF architecture is a layered architecture which have Managed, Unmanaged and Core API layers



WPF Architecture

1. Managed Layer

Managed layer has two main components – Presentation Framework and Presentation Core.

1. Presentation Framework provides the required functionalities that we need to build the WPF applications such as controls, data bindings, styling, shapes, media, documents, annotations, animation and more. PresentationFamework.dll is responsible for this purpose.
2. Presentation Core acts as a managed wrapper around MILCore and provides public interface for MIL. Presentation Core is the home for WPF Visual System and provides classes for creating application visual tree. The Visual System creates visual tree which contains applications Visual Elements and rendering instructions. PresentationCore.dll is responsible for this purpose.

2. Unmanaged Layer

This layer is also called milcore or Media Integration Library Core. MilCore is written in unmanaged code in order to enable tight integration with DirectX. DirectX engine is underlying technology used in WPF to display all graphics, allowing for efficient hardware and software rendering. MIL has Composition System that receives rendering instructions from Visual System and translates into data that can be understood by DirectX to render user interface.

3. Core API Layer

This layer has OS core components like Kernel, User32, GDI, Device Drivers, Graphic cards etc. These components are used by the application to access low level APIs. User32 manages memory and process separation.

4a	Using windows forms , write steps to work with mouse to draw on a form that explains mouse event handling.	10	CO3	L4

This event occurs when the user clicks the control with the mouse. The handler for this event receives an argument of type MouseEventArgs. Handle this event when you need to get information about the mouse when a click occurs.

```
private void Control1_MouseClick(Object sender, MouseEventArgs e) {

System.Text.StringBuilder messageBoxCS = new System.Text.StringBuilder();
messageBoxCS.AppendFormat("{0} = {1}", "Button", e.Button );
messageBoxCS.AppendLine();
messageBoxCS.AppendFormat("{0} = {1}", "Clicks", e.Clicks );
messageBoxCS.AppendLine();
messageBoxCS.AppendFormat("{0} = {1}", "X", e.X );
messageBoxCS.AppendLine();
messageBoxCS.AppendFormat("{0} = {1}", "Y", e.Y );
messageBoxCS.AppendLine();
messageBoxCS.AppendFormat("{0} = {1}", "Delta", e.Delta );
messageBoxCS.AppendLine();
messageBoxCS.AppendFormat("{0} = {1}", "Location", e.Location );
messageBoxCS.AppendLine();
MessageBox.Show(messageBoxCS.ToString(), "MouseClick Event" );
}

comboBox1.Items.RemoveAt(1);
```

5a	Explain XAML in detail.	10	CO4	L2

XAML is a new descriptive programming language developed by Microsoft to write user interfaces for next-generation managed applications. XAML is the language to build user interfaces for Windows and Mobile applications that use Windows Presentation Foundation (WPF), UWP, and Xamarin Forms.

The purpose of XAML is simple, to create user interfaces using a markup language that looks like XML. Most of the time, you will be using a designer to create your XAML but you're free to directly manipulate XAML by hand.

XAML uses the XML format for elements and attributes. Each element in XAML represents an object which is an instance of a type. The scope of a type (class, enumeration etc.) is defined as a namespace that physically resides in an assembly (DLL) of the .NET Framework library.
Similar to XML, a XAML element syntax always starts with an open angle bracket (<) and ends with a close angle bracket (>). Each element tag also has a start tag and an end tag. For example, a Button object is represented by the <Button> object element. The following code snippet represents a Button object element.

```
<Button></Button>
```

Alternatively, you can use a self-closing format to close the bracket.

```
<Button />
```

An object element in XAML represents a type. A type can be a control, a class or other objects

defined in the framework library.
The Root Elements

Each XAML document must have a root element. The root element usually works as a container and defines the namespaces and basic properties of the element. Three most common root elements are <Windows />, <Page />, and <UserControl >. The <ResourceDirectory /> and <Application /> are other two root elements that can be used in a XAML file.

The Window element represents a Window container. The following code snippet shows a Window element with its Height, Width, Title and x:Name attributes. The x:Name attribute of an element represents the ID of an element used to access the element in the code-behind. The code snippet also sets xmlns and xmlns:x attributes that represent the namespaces used in the code. The x:Class attribute represents the code-behind class name.

6a Explain steps in session tracking with http session state using cookies. 10 CO4 L4

A session is defined as the period of time that a unique user interacts with a Web application. Active Server Pages (ASP) developers who wish to retain data for unique user sessions can use an intrinsic feature known as session state.

Programmatically, session state is nothing more than memory in the shape of a dictionary or hash table, e.g. key-value pairs, which can be set and read for the duration of a user's session. For example, a user selects stocks to track and the Web application can store these values in the user's ASP session instance:

Session state settings in ASP.NET are configured through the ASP.NET XML configuration file config.web..

Config.web

There are two types of configuration files: a machine configuration file and an application configuration file, both named config.web. The two are identical, except that the machine configuration file applies settings to all applications but the application configuration files are either restrictive or expansive on an application-by-application basis.

- **Mode**. The mode setting supports three options: inproc, sqlserver, and stateserver. As stated earlier, ASP.NET supports two modes: in process and out of process. There are also two options for out-of-process state management: memory based (stateserver), and SQL Server based (sqlserver). We'll discuss implementing these options shortly.
- **Cookieless**. The cookieless option for ASP.NET is configured with this simple Boolean setting.
- **Timeout**. This option controls the length of time a session is considered valid. The session timeout is a sliding value; on each request the timeout period is set to the current time plus the timeout value.
- **Sqlconnectionstring**. The sqlconnectionstring identifies the database connection string that names the database used for mode sqlserver.
- **Server**. In the out-of-process mode stateserver, it names the server that is running the required Windows NT service: ASPState.
- **Port**. The port setting, which accompanies the server setting, identifies the port number that corresponds to the server setting for mode stateserver.

7a Explain the AJAX server controls in ASP.NET. 10 CO3 L3

AJAX stands for Asynchronous JavaScript and XML. This is a cross platform technology which speeds up response time. The AJAX server controls add script to the page which is executed and processed by the browser.

However like other ASP.NET server controls, these AJAX server controls also can have methods and event handlers associated with them, which are processed on the server side.

The ScriptManager Control

The ScriptManager control is the most important control and must be present on the page for other controls to work.

It has the basic syntax:

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
```

If you create an 'Ajax Enabled site' or add an 'AJAX Web Form' from the 'Add Item' dialog box, the web form automatically contains the script manager control. The ScriptManager control takes care of the client-side script for all the server side controls.

The UpdatePanel Control

The UpdatePanel control is a container control and derives from the Control class. It acts as a container for the child controls within it and does not have its own interface. When a control inside it triggers a post back, the UpdatePanel intervenes to initiate the post asynchronously and update just that portion of the page.

For example, if a button control is inside the update panel and it is clicked, only the controls within the update panel will be affected, the controls on the other parts of the page will not be affected. This is called the partial post back or the asynchronous post back.

The UpdateProgress Control

The UpdateProgress control provides a sort of feedback on the browser while one or more update panel controls are being updated. For example, while a user logs in or waits for server response while performing some database oriented job.

It provides a visual acknowledgement like "Loading page...", indicating the work is in progress.

The syntax for the UpdateProgress control is:

```
<asp:UpdateProgress ID="UpdateProgress1" runat="server" DynamicLayout="true"
AssociatedUpdatePanelID="UpdatePanel1" >

   <ProgressTemplate>
      Loading...
   </ProgressTemplate>

</asp:UpdateProgress>
```

The Timer Control

The timer control is used to initiate the post back automatically. This could be done in two ways:

(1) Setting the Triggers property of the UpdatePanel control:

```
<Triggers>
   <asp:AsyncPostBackTrigger ControlID="btnpanel2" EventName="Click" />
</Triggers>
```

8a Write a web based application to check the username and password entered through a web form are valid or not by checking with a database table.	10	CO4	L3

Page.aspx

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>Login Form</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<table>
<tr>
<td>
Username:
</td>
<td>
<asp:TextBox ID="txtUserName" runat="server"/>
<asp:RequiredFieldValidator ID="rfvUser" ErrorMessage="Please enter
Username" ControlToValidate="txtUserName" runat="server" />
</td>
</tr>
<tr>
<td>
Password:
</td>
<td>
<asp:TextBox ID="txtPWD" runat="server" TextMode="Password"/>
<asp:RequiredFieldValidator ID="rfvPWD" runat="server" ControlToValidate="txtPWD" ErrorMessage="Please
enter Password"/>
</td>
</tr>
<tr>
<td>
</td>
<td>
<asp:Button ID="btnSubmit" runat="server" Text="Submit" onclick="btnSubmit_Click" />
</td>
</tr>
</table>
</div>
</form>
</body>
</html>
```

Page.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
protected void btnSubmit_Click(object sender, EventArgs e)
{
SqlConnection con
= new SqlConnection(ConfigurationManager.ConnectionStrings["dbconnection"].ConnectionString);
con.Open();
SqlCommand cmd = new SqlCommand("select * from UserInformation where UserName =@username and
Password=@password",con);
cmd.Parameters.AddWithValue("@username", txtUserName.Text);
cmd.Parameters.AddWithValue("@password", txtPWD.Text);
SqlDataAdapter da = new SqlDataAdapter(cmd);
DataTable dt = new DataTable();
da.Fill(dt);
if(dt.Rows.Count>0)
{
Response.Redirect("Details.aspx");
}
else
{
ClientScript.RegisterStartupScript(Page.GetType(), "validation", "<script
language='javascript'>alert('Invalid Username and Password')</script>");
}
}
```