VII SEMESTER B.TECH INTERNAL EXAMINATIONS SEPTEMBER 2019
SUBJECT: INTERNET TECHNOLOGIES (CSE 4101) TEST 1 SCHEME
Date of Exam: 06-09-2019 03:00 PM – 04:00 PM Max. Marks: 15

1a. An object variable can be cast into which of these:
1) Itself 2) An interface that it supports
3) Base class from which it inherits 4) Some other class from same namespace
a) 1,2,3 b) 2,3,4 c) 1,3,4 d) 1,2,4 **0.5**

1b. What will be the output of the following error free code snippet:

```
using System;
public class Program{
    static void arrayMethod(int[] a){
        int[] b = new int[5];
    a = b;
    }
    public static void Main(string[] args){
        int[] arr = new int[10];
    arrayMethod(arr);
    Console.WriteLine(arr.Length);
    }
}
```

a) 5 **b) 10** c) 15 d) 20 **0.5**

1c. For which of the following control, the postback property is always true?
a) TextBox b) DropDownList **c) Button** d) CheckBoxList **0.5**

1d. The **_____** section of the web.config file allows you to define the connection information for accessing a database.
a) <connectionStrings> b) <system.web> c) <appSettings> d) <compilation> **0.5**

1e. What is NOT true about assemblies
a) Some type of Assemblies have .exe extension **b) logical grouping of code**
c) physical package to distribute code d) Some type of Assemblies have .dll extension **0.5**

2a. _____ system specially defines the rules for data types such as strings, numbers and arrays that are shared in all .NET languages.
a) CLR b) CLS **c) CTS** d) JIT **0.5**

2b. Pick the odd one with respect to web controls.
a) Grid View b) Calendar c) Validation Controls **d) Span** **0.5**

2c. _____ data types are allowed in Viewstate collection.
a) All .NET data type **b) All serializable .NET data type** c) Numeric d) Limited String **0.5**

2d. _____ property of the HttpCookie is used to delete the cookie.
a) Request **b) Expires** c) Response d) Purge **0.5**

2e. Which statement is false about Session state?
a) Times out after a predefined period
b) **All .NET data types are allowed if one uses out of process storage mode**
c) The scope is the whole ASP.NET application, and the data is global for the current user
d) Very secure, because data is never transmitted to the client **0.5**

3 Create a class called "**Calculation**" which has two automatic properties:
1. Result - integer to store the result of calculation
2. Compute – function reference to a method which takes two integer arguments and returns nothing.
Consider **WebPage1** with a dropdown and a button as shown in Fig.Q.3.1.
Create an instance of **Calculation** class in the code behind of the **WebPage1**. The webpage should contain two methods Add and Subtract, which should perform the appropriate operation and assign the result to the **Result** property of the **Calculation** object. On button click, assign appropriate method to **Compute** property of the **Calculation** object based on selection in the dropdown and pass the calculation object to **WebPage2** using Session State.
Consider **WebPage2** with 2 textboxes, label and a button as shown in Fig.Q.3.2. On button click, retrieve the **Calculation** object from Session and execute the method referenced by **Compute** property of the **Calculation** object using the values entered in textboxes and display the result in label. **3**

NOTE: ONLY write the **Calculation** class and code behind files of **WebPage1** and **WebPage2.**

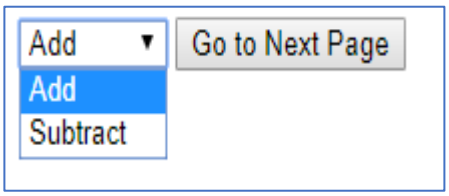


Fig.Q.3.1

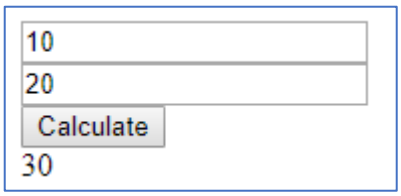


Fig.Q.3.2

Solution

```
//0.5 marks
public delegate void Computation(int a, int b);
public class Calculation
{
   //0.5 marks
   public int Result { get; set; }
   public Computation Compute { get; set; }
}
```

Webpage1

```
public partial class _Default : System.Web.UI.Page
{
   Calculation c = new Calculation();
   protected void Button1_Click(object sender, EventArgs e)
   {
      //0.5 marks
      switch(DropDownList1.SelectedIndex)
      {
         case 0:
            c.Compute += Add;
            break;
         case 1:
            c.Compute += Subtract;
            break;
      }
      //0.5 marks
      Session["Calculation"] = c;
      Response.Redirect("Default2.aspx");
   }

   private void Add(int a, int b)
   {
      c.Result = a + b;
   }

   private void Subtract(int a, int b)
   {
      c.Result = a - b;
   }
}
```

Webpage2

```
   protected void Page_Load(object sender, EventArgs e)
   {
      //0.5 marks
      Calculation c = Session["Calculation"] as Calculation;

      if(c != null)
```

```
    {
      //0.5 marks
      int a = Convert.ToInt32(TextBox1.Text);
      int b = Convert.ToInt32(TextBox2.Text);
      c.Compute(a, b);
      Label1.Text = c.Result.ToString();
    }

  }
```

4. With the help of examples, explain optional and named parameters in C#. What are the constraints on optional parameters?

An *optional parameter* is any parameter that has a default value. If your method has normal parameters and optional parameters, the optional parameters must be placed at the end of the parameter list. Here's an example of a method that has a single optional parameter:

```
private string GetUserName(int ID, bool useShortForm = false)
{
// Code here.
}
```

Here, the useShortForm parameter is optional, which gives you two ways to call the GetUserName() method:

```
// Explicitly set the useShortForm parameter.
name = GetUserName(401, true);
// Don't set the useShortForm parameter, and use the default value (false).
name = GetUserName(401);
```

Sometimes you'll have a method with multiple optional parameters, like this one:

```
private decimal GetSalesTotalForRegion(int regionID, decimal minSale = 0,
decimal maxSale = Decimal.MaxValue, bool includeTax = false)
{
// Code here.
}
```

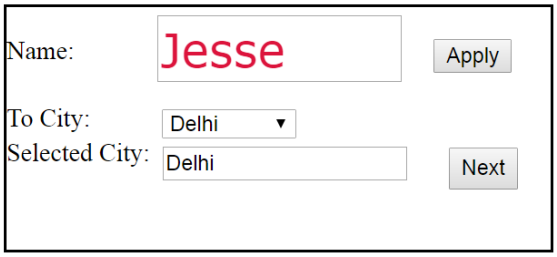
In this situation, the easiest option is to pick out the parameters you want to set by name. This feature is called *named parameters*, and to use it you simply add the parameter name followed by a colon (:), followed by
the value, as shown here:

```
total = GetSalesTotalForRegion(523, maxSale: 5000);
```

Scheme: Optional Parameter with default value →0.5M
Constraint on optional parameter →0.5M
Named Parameter when multiple optional parameters→0.5M
: operator to assign to named parameter→0.5M

2

5. Create a Web form as show in the figure below. Web form should contain only web controls for user input.



***** When *"Apply"* button is clicked, the forecolor, font-name and font-size of the first textbox(Name) should be set to "Crimson", "Verdana" and 22 respectively in code behind (*.cs)* file. Mention the required namespaces.

***** Using Page_Load method, load names of at least 3 different cities in the DropDownList control. When user selects a city, the selected city name should be displayed in the second textbox (*Selected City)* immediately.

3

***** When *"Next"* button is clicked, send the user to a new page*("NewPage.aspx")* without involving the browser.
(**Note:** Write the entire code behind file and only the aspx tag of DropDownList control)

Solution:

```
<asp:DropDownList   ID="DDList"   runat="server"
OnSelectedIndexChanged="DDList_SelectedIndexChanged"
AutoPostBack="true" ></asp:DropDownList>                    (0.5M)

using System;
public partial class Test
{
protected void Page_Load(object sender, EventArgs e)
        {
        if (!this.IsPostBack)
        {
                DDList.Items.Add("Bangalore");                    (0.5M)
                DDList.Items.Add("Mumbai");
                DDList.Items.Add("Delhi");
        }
        }
protected void Apply_Click(object sender, EventArgs e)
{
     Name.ForeColor = System.Drawing.Color.Crimson;      `     (1M)
     Name.Font.Name = "Verdana";
     Name.Font.Size = 22;
}
protected void DDList_SelectedIndexChanged(object sender, EventArgs e)
{
        Selected_City.Text = DDList.SelectedItem.Text;            (0.5M)
}
protected void NewPage_Click(object sender, EventArgs e)
{
        Server.Transfer("NewPage.aspx");                          (0.5M)
} }
```

6. Explain enumeration in C# with a code snippet
(Description 1 Marks + Example 1 Marks)

- An enumeration is a group of related constants, each of which is given a descriptive name.
- Each value in an enumeration corresponds to a preset integer.
- In your code, however, you can refer to an enumerated value by name, which makes your code clearer and helps prevent errors.
- Example

```
//Define an enumeration called UserType with three possible values. (0.5M)
        enum UserType
        {
                Admin,
                Guest,
                Invalid
         }
//Create a new value and set it equal to the UserType.Admin constant. (0.5 M)
        UserType newUserType;
        newUserType = UserType.Admin
```

- Internally, **enumerations are maintained as numbers**. In the preceding example, 0 is automatically assigned to Admin, 1 to Guest, and 2 to Invalid.