PART-A
(Each question carries 2 marks, Answer any FIVE questions, Q.No. 8 – Compulsory)

1. What is meant by managed code.
   Managed Code in Microsoft .Net Framework, is the code that has executed by the Common Language Runtime (CLR) environment. On the other hand Unmanaged Code is directly executed by the computer's CPU.

2. How constants are created is in C#?
   Constants are immutable values which are known at compile time and do not change for the life of the program. Constants are declared with the const modifier
   
   ```
   class Calendar1
   {
       public const int months = 12;
   }
   ```

3. Differentiate accessing key and shortcut keys.
   **Access keys**
   Access keys consist of Alt + alphanumeric key(s), are mostly for accessibility purposes, are assigned to all menus and most dialog box controls, are not meant to be memorized, affect only the current window, and are localized.

   **Shortcut keys**
   Shortcut keys mostly use Control (Ctrl) and Function (Fn) key sequences, are designed more for advanced users, are assigned only to the most often used commands, are intended to be memorized and are documented only in menus, tooltips, and help, must be assigned with consistency because they are meant to be memorized, and are not localized.

4. Write a C# program to create and handle the button click event.
   ```
   using System;
   using System.Windows.Forms;
   namespace WindowsFormsApplication21
   {
       public partial class Form1 : Form 
       {
           public Form1()
           {
               InitializeComponent();
           }
           private void button1_Click(object sender, EventArgs e)
           {
               MessageBox.Show("This is a test message");
           }
       }
   }
   ```

5. Explain connection object with example.
   The Connection object
   Listed below are the common connection object methods we could work with:
   A. Open - Opens the connection to our database
   B. Close - Closes the database connection
   C. Dispose - Releases the resources on the connection object. Used to force garbage collecting, ensuring no resources are being held after our connection is used.
   D. State - Tells you what type of connection state your object is in, often used to check whether your connection is still using any resources.
   Ex. if (ConnectionObject.State == ConnectionState.Open)
6. Write about notations in DTD.

NOTATION IN DTD

A Notations provide information about the format of external entities (non-XML) files such as an image or video file that are effectively included into a source XML file. This allows the XML Parser or client application to correctly process the data contained. The notation syntax takes the following forms

<!NOTATION Name PUBLIC PublicID>

7. What is XML schema?

XML Schema is commonly known as XML Schema Definition (XSD). It is used to describe and validate the structure and content of XML data. XML schema defines the elements, attributes and data types. Schema element supports Namespaces. It is similar to a database schema that describes the data in a database.

8. Explain decision making statement in C# .NET.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>if statement</td>
<td>An if statement consists of a boolean expression followed by one or more statements.</td>
</tr>
<tr>
<td>if...else statement</td>
<td>An if statement can be followed by an optional else statement, which executes when the boolean expression is false.</td>
</tr>
<tr>
<td>nested if statements</td>
<td>You can use one if or else if statement inside another if or else if statement(s).</td>
</tr>
<tr>
<td>witch statement</td>
<td>A switch statement allows a variable to be tested for equality against a list of values.</td>
</tr>
<tr>
<td>nested switch statements</td>
<td>You can use one switch statement inside another switch statement(s)</td>
</tr>
</tbody>
</table>

PART – B

(Each question carries 3 marks, Answer any FIVE questions. Q.No 16 – Compulsory)

9. Explain the steps involved in automatic memory management.

Automatic Memory Management

The .NET framework has introduced a concept called Garbage Collector. This mechanism keeps track of the allocated memory references and releases the memory when it is not in reference. Since it is automatic it relieves the programmer to manage unused allocated memory. This concept is called Automatic Memory Management.

10. What is the difference between class and interface?

- A Class has both definition and an implementation whereas Interface only has a definition.
- A Class can be instantiated but an Interface cannot be instantiated you can create an instance of an Object that implements the Interface.
- A Class is a full body entity with members, methods along with their definition and implementation. An Interface is just a set of definition that you must implement in your Class inheriting that Interface.

11. What is the use of THIS reference?

The "this" keyword is a special type of reference variable that is implicitly defined within each constructor and non-static method as a first parameter of the type class in which it is defined.

"this" in C# is used to access the field variables. All the field variables must be accessed using "this" keyword.

For example, consider the following class written in C#.
class Demo {
    int a=2,b=10;
    public void Get()
    {
        int a=23,b=34;
        Console.WriteLine("a={0} b={1}",this.a,this.b);
        Console.WriteLine("It is the class variable");
        Console.WriteLine("Now the local variables are:");

        Console.WriteLine("a={0} b={1}",a,b);
    }
}

class MainClass {
    static void Main(string args[]) {
        Demo d= new Demo();
        d.Get();
    }
}

12. What is an event? List mouse events.
An event in C# is a way for a class to provide notifications to clients of that class when some interesting thing happens to an object. The most familiar use for events is in graphical user interfaces; typically, the classes that represent controls in the interface have events that are notified when the user does something to the control (for example, click a button).

    Mousemove
    Mousedown
    Mouseup
    Mouseclick

13. Explain the steps, in connecting to data source with example.
We need to perform four major tasks:
- Connecting to the database
- Passing the request to the database, i.e., a command like select, insert, or update.
- Getting back the results, i.e., rows and/or the number of rows effected.
- Storing the result and displaying it to the user.

This can be visualized as:

14. Compare the performance of data reader and dataset.
DataReader
The ADO.NET DataReader is used to retrieve read-only(cannot update data back to datasource) and forward-only(cannot read backward/random) data from a database.
- DataReader increases application performance and reduces system overheads. This is due to one row at a time is stored in memory.
You create a DataReader by calling Command. ExecuteReader after creating an instance of the Command object.

This is a connected architecture: The data is available as long as the connection with database exists.

You need to open and close the connection manually in code.

**DataSet**

- The DataSet is an in-memory representation of data.
- It can be used with multiple data sources. That is a single DataSet can hold the data from different data sources holding data from different databases/tables.
- The DataSet represents a complete set of data including related tables, constraints, and relationships among the tables.
- The DataSet can also persist and reload its contents as XML and its schema as XML Schema definition language (XSD) schema.
- The DataAdapter acts as a bridge between a DataSet and a data source for retrieving and saving data.
- The DataAdapter helps mapping the data in the DataSet to match the data in the data source.
- Also, Upon an update of dataset, it allows changing the data in the data source to match the data in the DataSet.
- No need to manually open and close connection in code.
- Hence, point (8) says that it is a disconnected architecture. Fill the data in DataSet and that's it. No connection existence required.

15. **Explain element declaration in XML with example.**

**Elements**

XML elements can be defined as building blocks of an XML. Elements can behave as containers to hold text, elements, attributes, media objects or all of these. Each XML document contains one or more elements, the scope of which are either delimited by start and end tags, or for empty elements, by an empty-element tag.

Following is the syntax to write an XML element:
```xml
<element-name attribute1 attribute2>
.....content
</element-name>
```

where

- element-name is the name of the element. The *name* its case in the start and end tags must match.
- attribute1, attribute2 are attributes of the element separated by white spaces. An attribute defines a property of the element. It associates a name with a value, which is a string of characters. An attribute is written as:
  ```xml
  name = "value"
  ```
  name is followed by an = sign and a string *value* inside double(" ") or single(‘’) quotes.

16. **Explain JIT compilation process.**

Metadata in .Net is binary information which describes the characteristics of a resource. This information include Description of the Assembly , Data Types and members with their declarations and implementations, references to other types and members , Security permissions etc. A module's metadata contains everything that needed to interact with another module.

During the compile time Metadata created with Microsoft Intermediate Language (MSIL) and stored in a file called a Manifest. Both Metadata and Microsoft Intermediate Language (MSIL) together wrapped in a Portable Executable (PE) file. During the runtime of a program Just In Time (JIT) compiler of the Common Language Runtime (CLR) uses the Metadata and converts Microsoft Intermediate Language (MSIL) into native code. When code is executed, the runtime loads metadata into memory and references it to discover information about your code's classes, members, inheritance, and so on. Moreover Metadata eliminating the need for Interface Definition Language (IDL) files, header files, or any external method of component reference.
17.(a) Explain the .NET framework architecture with a neat diagram [10]

The .NET Framework is Microsoft's Managed Code programming model for building applications on Windows clients, servers, and mobile or embedded devices.

The Microsoft .Net Framework is a platform that provides tools and technologies you need to build Networked Applications as well as Distributed Web Services and Web Applications. The .Net Framework provides the necessary compile time and run-time foundation to build and run any language that conforms to the Common Language Specification (CLS). The main two components of .Net Framework are Common Language Runtime (CLR) and .Net Framework Class Library (FCL).

Managed codes and CLR

Managed Code in Microsoft .Net Framework, is the code that has executed by the Common Language Runtime (CLR) environment. On the other hand Unmanaged Code is directly executed by the computer's CPU. Data types, error-handling mechanisms, creation and destruction rules, and design guidelines vary between managed and unmanaged object models.

The benefits of Managed Code include programmers convenience and enhanced security. Managed code is designed to be more reliable and robust than unmanaged code, examples are Garbage Collection, Type Safety etc. The Managed Code running in a Common Language Runtime (CLR) cannot be accessed outside the runtime environment as well as cannot call directly from outside the runtime environment. This makes the programs more isolated and at the same time computers are more secure. Unmanaged Code can bypass the .NET Framework and make direct calls to the Operating System. Calling unmanaged code presents a major security risk.

CLR

The Common Language Runtime (CLR) is a an Execution Environment. Common Language Runtime (CLR)'s main tasks are to convert the .NET Managed Code to native code, manage running code like a Virtual Machine and also controls the interaction with the Operating System.

Common Language Runtime (CLR) manages Thread executions, Memory Management that is allocation of Objects and Buffers, Garbage Collection (GC) - Clean up the unused Objects and buffers, Exception Handling, Common Type System (CTS) that is all .NET language that conforms to the Common Language Specification (CLS) have the same primitive Data Types, Code safety verifications - code can be verified to ensure type safety, Language integration that is Common Language Runtime (CLR) follow a set of specification called Common Language Specification (CLS), this will ensure the interoperability between languages, Integrated security and other system services.

Intermediate Languages

Intermediate language (IL) is an object-oriented programming language designed to be used by compilers for the .NET Framework before static or dynamic compilation to machine code. The IL is used by the .NET Framework to generate machine-independent code as the output of compilation of the source code written in any .NET programming language.

IL is a stack-based assembly language that gets converted to bytecode during execution of a virtual machine. It is defined by the common language infrastructure (CLI) specification. As IL is used for automatic generation of compiled code, there is no need to learn its syntax. This term is also known as Microsoft intermediate language (MSIL) or common intermediate language (CIL).

Metadata and JIT Compilation
Metadata in .Net is binary information which describes the characteristics of a resource. This information include Description of the Assembly, Data Types and members with their declarations and implementations, references to other types and members, Security permissions etc. A module's metadata contains everything that needed to interact with another module.

During the compile time Metadata created with Microsoft Intermediate Language (MSIL) and stored in a file called a Manifest. Both Metadata and Microsoft Intermediate Language (MSIL) together wrapped in a Portable Executable (PE) file. During the runtime of a program Just In Time (JIT) compiler of the Common Language Runtime (CLR) uses the Metadata and converts Microsoft Intermediate Language (MSIL) into native code. When code is executed, the runtime loads metadata into memory and references it to discover information about your code's classes, members, inheritance, and so on. Moreover Metadata eliminating the need for Interface Definition Language (IDL) files, header files, or any external method of component reference.

**Automatic Memory Management**

The .NET framework has introduced a concept called Garbage Collector. This mechanism keeps track of the allocated memory references and releases the memory when it is not in reference. Since it is automatic it relieves the programmer to manage unused allocated memory. This concept is called Automatic Memory Management.

---

17. (b) (i). Briefly explain solution explore window of visual studio IDE [5]

In Integrated Development Environment (IDE) is software that facilitates application development. In the context of .NET-based applications, Visual Studio is the most commonly used IDE. Some of the key features included are:

- Single IDE for all .NET applications. Therefore no switching required to other IDEs for developing .NET applications
- Single .NET solution for an application which has been built on code written in multiple languages
- Code editor supporting Intelligence and code refactoring
- Compilation from within the environment based on defined configuration options
- Integrated debugger that works at source and machine level
- Plug-in architecture that helps to add tools for domain specific languages
- Customizable environment to help the user to configure the IDE based on the required settings
- Browser that is built-in within the IDE helps to view content from internet such as help, source-code, etc. in online mode.

17.(b)(ii) Write short notes on .NET objects. [5]

The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object oriented, providing types from which your own managed code can derive functionality. This not only makes the .NET Framework types easy to use, but also reduces the time associated with learning new features of the .NET Framework. For example, the .NET Framework collection classes implement a set of interfaces that you can use to develop your own collection classes. Your collection classes will blend seamlessly with the classes in the .NET Framework.

18. (a) Explain the various looping statements in C# .NET with example. [10]

In Loop Statements there may be a situation, when you need to execute a block of code several number of times. In general, the statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.

Programming languages provide various control structures that allow for more complicated execution paths. A loop statement allows us to execute a statement or a group of statements multiple times and following is the general form of a loop statement in most of the programming languages:
**While Loop**
A while loop statement in C# repeatedly executes a target statement as long as a given condition is true.

Syntax
while(condition)
{
    statement(s);
}

Here, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is any non-zero value. The loop iterates while the condition is true. When the condition becomes false, program control passes to the line immediately following the loop. When the condition is tested and the result is false, the loop body is skipped and the first statement after the while loop is executed.

**For Loop**
A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax
for ( init; condition; increment )
{
    statement(s);
}

Example:
using System;
namespace Loops
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int a = 10; a < 20; a = a + 1)
            {
                Console.WriteLine("value of a: \{0\}", a);
            }
            Console.ReadLine();
        }
    }
}

**Do...While Loop**
Unlike for and while loops, which test the loop condition at the start of the loop, the do...while loop checks its condition at the end of the loop. A do...while loop is similar to a while loop, except that a do...while loop is guaranteed to execute at least one time. It is also known as at least once loop.

Syntax
Do
{
    statement(s);
}
while( condition );

Notice that the conditional expression appears at the end of the loop, so the statement(s) in the loop execute once before the condition is tested. If the condition is true, the flow of control jumps back up to do, and the statement(s) in the loop execute again. This process repeats until the given condition becomes false.

**Foreach**
Foreach loop is a different kind of looping constructs in C# programming that doesn’t includes initialization, termination and increment/decrement characteristics. It uses collection to take value one by one and then processes them.

With a foreach loop, we evaluate each element individually. An index is not needed. With no indexes, loops are easier to write and programs are simpler.

Syntax:
foreach (string name in arr)
{
}

**Nested Loops**

C# allows to use one loop inside another loop. Following section shows few examples to illustrate the concept.

Syntax:
The syntax for a nested for loop statement in C# is as follows:
for ( init; condition; increment )
{
 for ( init; condition; increment )
{
 statement(s);
 }
 statement(s);
}
The syntax for a nested while loop statement in C# is as follows:
while(condition)
{
 while(condition)
{
 statement(s);
 }
 statement(s);
}
The syntax for a nested do...while loop statement in C# is as follows:
do
{
 statement(s);
 do
{
 statement(s);
 }
 while( condition );
 } while( condition );

(OR)

**18.(b)(i) Define array. Differentiate regular and jagged arrays.**[5]

**Arrays**

An array stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type stored at contiguous memory locations.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index. All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

**Jagged Arrays**
A Jagged array is an array of arrays.
You can declare a jagged array named scores of type int as:
```csharp
int[][] scores = new int[5][];
for (int i = 0; i < scores.Length; i++)
{
    scores[i] = new int[4];
}
```
You can initialize a jagged array as:
```csharp
int[][] scores = new int[2][]
{
    new int[]{92,93,94},
    new int[]{85,66,87,88});
```
Where, scores is an array of two arrays of integers - scores[0] is an array of 3 integers and scores[1] is an array of 4 integers.

18.(b)(ii). Write short notes on structures and enumeration [5]

**Structures:**
In C#, a structure is a value type data type. It helps you to make a single variable hold related data of various data types. The struct keyword is used for creating a structure. Structures are used to represent a record. Suppose you want to keep track of your books in a library. You might want to track the following attributes about each book:
Title
Author
Subject
Book ID
To define a structure, you must use the struct statement. The struct statement defines a new data type, with more than one member for your program. For example, here is the way you can declare the Book structure:
```csharp
struct Books
{
    public string title;
    public string author;
    public string subject;
    public int book_id;
};
```

**Enumeration**
An enumeration is a set of named integer constants. An enumerated type is declared using the enum keyword. C# enumerations are value data type. In other words, enumeration contains its own values and cannot inherit or cannot pass inheritance.
The general syntax for declaring an enumeration is:
```csharp
enum <enum_name>
{
    enumeration list
};
```
Where, the enum_name specifies the enumeration type name. The enumeration list is a comma-separated list of identifiers. Each of the symbols in the enumeration list stands for an integer value, one greater than the symbol that precedes it. By default, the value of the first enumeration symbol is 0.

19.(a) Explain with example the steps to create a menu and context menu. [10]
You can add menus to Windows Forms at design time by adding the MainMenu component and then appending menu items to it using the Menu Designer.
Adding Menu, Menu Items to a Menu

First add a MainMenu control to the form. Then to add menu items to it add MenuItem objects to the collection. By default, a MainMenu object contains no menu items, so that the first menu item added becomes the menu heading. Menu items can also be dynamically added when they are created, such that properties are set at the time of their creation and addition.

If you need a separator bar, right click on your menu then go to insert->Separator.

Context menus (Popup menus)
Context menus are used inside applications to provide users access to often used commands by means of a right-click of the mouse. Often, context menus are assigned to controls, and provide particular commands that relate to that precise control.

(OR)

19.(b)(i). Write the steps to create MDI child window with example. [10]
Step to Create and Implement MDI Child Form
1. Assumes there is an MDI parent form having MenuStrip with option New, Window and Close in New Menu, main form contain one Child form having a RichTextBox.Add one more control in Main Form MenuStrip as Cascade Windows.
2. Double click on Cascade Windows control and write this Code.
3. private void cascadeWindowToolStripMenuItem_Click(object sender, EventArgs e) { this.LayoutMdi(System.Windows.Forms.MdiLayout.Cascade); }
4. Debug the application and click on New button two times then two MDI Child form with RichTextBox will open. Now by using Cascade Windows control in the Main Menu you can arrange all the opened Mdi Child Form in Cascade mode.
19. (b) (ii). Write short notes on progress bar and month calendar. [5]

**ProgressBar Control**

A progress bar is a control that an application can use to indicate the progress of a lengthy operation such as calculating a complex result, downloading a large file from the Web etc.

ProgressBar controls are used whenever an operation takes more than a short period of time. The Maximum and Minimum properties define the range of values to represent the progress of a task.

- **Minimum**: Sets the lower value for the range of valid values for progress.
- **Maximum**: Sets the upper value for the range of valid values for progress.
- **Value**: This property obtains or sets the current level of progress.

By default, Minimum and Maximum are set to 0 and 100. As the task proceeds, the ProgressBar fills in from the left to the right. To delay the program briefly so that you can view changes in the progress bar clearly.

![](image)

**MonthCalendar control**

MonthCalendar is a selectable calendar widget. On the MonthCalendar, a user can select a day, or a range of days. The user can also scroll through the months. This control provides many useful options. It is ideal for instant calendars.

The MonthControl provides two important properties of the calendar called MaxDate and MinDate. These indicate the maximum and minimum selectable dates. These dates give you a lot of range to select dates.

- **Date Properties**: MinDate: 1/1/1753 MaxDate: 12/31/9998
- **ShowToday**: is by default set to true. If you set it to false, it will not be present at the bottom of the calendar.
- **ShowTodayCircle**: property adjusts the visibility of the box on the left of the "Today" display.

The MonthCalendar provides an event-driven user interface and you can provide and hook up event handlers to execute code on user actions. The DateChanged event allows you to detect whenever the user changes the date to something else.

![](image)

20. (a) What is stored procedure? Write the steps to create stored procedure with example. [10]

Stored Procedures are a set of sql commands which are compiled and are stored inside the database.

Sample application Using a Stored Procedure with a Command.
Creating a sample application using a Stored Procedure with a Command object then we need to specify it as. Initially create a object of SqlConnection class which is available in System.Data.SqlClient namespace. Open the connection using the Open() method.

```csharp
SqlCommand command = new SqlCommand("RegionUpdate", con);
command.CommandType = CommandType.StoredProcedure;

command.Parameters.Add(new SqlParameter("@RegionID", SqlDbType.Int, 0, "RegionID"));
command.Parameters.Add(new SqlParameter("@RegionDescription", SqlDbType.NChar, 50, "RegionDescription"));

command.Parameters[0].Value = 4;
command.Parameters[1].Value = "SouthEast";
```

Create the following stored procedure on the Region table in the Northwind database which accepts two parameters and does not have any output parameters.

```sql
CREATE PROCEDURE RegionUpdate (@RegionID INTEGER, @RegionDescription NCHAR(50)) AS
SET NOCOUNT OFF
UPDATE Region SET RegionDescription = @RegionDescription
```

Create a SqlCommand object with the parameters as the name of the stored procedure that is to be executed and the connection object con to which the command is to be sent for execution.

```csharp
SqlCommand command = new SqlCommand("RegionUpdate", con);
command.CommandText = "RegionUpdate";
command.CommandType = CommandType.StoredProcedure;

command.Parameters.Add(new SqlParameter("@RegionID", SqlDbType.Int, 0, "RegionID"));
command.Parameters.Add(new SqlParameter("@RegionDescription", SqlDbType.NChar, 50, "RegionDescription"));

command.Parameters[0].Value = 4;
command.Parameters[1].Value = "SouthEast";
```

20.(b).(i) Define data adapter. Explain with a program how data adapter works together with dataset. [5]

Using an adapter, you can read, add, update, and delete records in a data source. To allow you to specify how each of these operations should occur, an adapter supports the following four properties:
- **Select Command** – reference to a command that retrieves rows from the data store.
- **Insert Command** – reference to a command for inserting rows into the data store.
- **Update Command** – reference to a command for modifying rows in the data store.
- **Delete Command** – reference to a command for deleting rows from the data store.

```csharp
SqlDataAdapter daEmp = new SqlDataAdapter("SELECT empno, empname, empaddress FROM EMPLOYEE", conn);
//Create a DataSet Object
DataSet dsEmp = new DataSet();
//Fill the DataSet daEmp.Fill(dsEmp,"EMPLOYEE");
//Let us print first row and first column of the table
Console.WriteLine(dsEmp.Tables["EMPLOYEE"].Rows[0][0].ToString());
//Assign a value to the first column dsEmp.Tables["EMPLOYEE"].Rows[0][0] = "12345"; //This will generate runtime error as empno column is integer
```

20.(b)(ii) Explain the object model of dataset. [5]

The ADO.NET object model consists of two key components as follows:
- **Connected model** (.NET Data Provider - a set of components including the Connection, Command, DataReader, and DataAdapter objects): We have the control over the database connection, so we have to explicitly open/close the objects in model have to directly talk to the database and hence are database specific Connection, Command and Data Reader are members of this set.
Disconnected model (DataSet): It’s complimentary to earlier model in the sense that the object itself decides when the connection will be opened and closed. We don’t have to do it implicitly as a result only one component of this model which talks to the database directly is data adapter whereas the cache which contains the data never speaks to the database directly and isn’t database specific.

21.(a) How will you declare elements and entities in DTD? Explain with example. [10]

The document type (DOCTYPE) declaration consists of an internal, or references an external Document Type Definition (DTD). It can also have a combination of both internal and external DTDs. The DTD defines the constraints on the structure of an XML document. It declares all of the document’s element types, children element types, and the order and number of each element type. It also declares any attributes, entities, notations, processing instructions, comments, and PE references in the document.

The Internal DTD:

```xml
<!DOCTYPE root_element [ Document Type Definition (DTD):
  elements/attributes/entities/notations/
  processing instructions/comments/PE references ]>
```

Example: `<!xml version="1.0" standalone="yes" ?>
<!--open the DOCTYPE declaration - the open square bracket indicates an internal DTD-->
<!DOCTYPE foo [ <!--define the internal DTD--> <!ELEMENT foo (#PCDATA)> <!--close the DOCTYPE declaration--> ]>
<foo>Hello World.</foo>

(OR)

21.(b) (i) Define serialization. Explain XML serialization architecture. [5]

Serialization is the process of converting an object into a form that can be readily transported. For example, you can serialize an object and transport it over the Internet using HTTP between a client and a server. On the other end, deserialization reconstructs the object from the stream. XML serialization serializes only the public fields and property values of an object into an XML stream. XML serialization does not include type information. For example, if you have a Book object that exists in the Library namespace, there is no guarantee that it is deserialized into an object of the same type.

Advantages of Using XML Serialization

The XmlSerializer class gives you complete and flexible control when you serialize an object as XML. If you are creating an XML Web service, you can apply attributes that control serialization to classes and members to ensure that the XML output conforms to a specific schema. For example, XmlSerializer enables you to:

- Specify whether a field or property should be encoded as an attribute or an element.
- Specify an XML namespace to use.
- Specify the name of an element or attribute if a field or property name is inappropriate.

Another advantage of XML serialization is that you have no constraints on the applications you develop.

21.(b) (ii) Define SOAP. Explain SOAP building blocks with example [5]

SOAP is an acronym for Simple Object Access Protocol. It is an XML-based messaging protocol for exchanging information among computers. SOAP is an application of the XML specification.

SOAP message is an ordinary XML document containing the following elements –

- **Envelope** – Defines the start and the end of the message. It is a mandatory element.
- **Header** – Contains any optional attributes of the message used in processing the message, either at an intermediary point or at the ultimate end-point. It is an optional element.
- **Body** – Contains the XML data comprising the message being sent. It is a mandatory element.
Fault – An optional Fault element that provides information about errors that occur while processing the message.