DOT NET Interview Questions and Answers --

Contents

.NET Framework

1. What is .NET Framework?

.NET Framework is a complete environment that allows developers to develop, run, and deploy the following applications:

- Console applications
- Windows Forms applications
- Windows Presentation Foundation (WPF) applications
- Web applications (ASP.NET applications)
- Web services
- Windows services
- Service-oriented applications using Windows Communication Foundation (WCF)
- Workflow-enabled applications using Windows Workflow Foundation (WF)

.NET Framework also enables a developer to create sharable components to be used in distributed computing architecture. NET Framework supports the object-oriented programming model for multiple languages, such as Visual Basic, Visual C#, and Visual C++. .NET Framework supports multiple programming languages in a manner that allows language interoperability. This implies that each language can use the code written in some other language.

2. What are the main components of .NET Framework?

.NET Framework provides enormous advantages to software developers in comparison to the advantages provided by other platforms. Microsoft has united various modern as well as existing technologies of software development in .NET Framework. These technologies are used by developers to develop highly efficient applications for modern as well as future business needs. The following are the key components of .NET Framework:

- .NET Framework Class Library
- Common Language Runtime
- Dynamic Language Runtimes (DLR)
- Application Domains
- Runtime Host
- Common Type System
- Metadata and Self-Describing Components
- Cross-Language Interoperability
- .NET Framework Security
- Profiling
- Side-by-Side Execution

3. List the new features added in .NET Framework 4.0.

The following are the new features of .NET Framework 4.0:

- Improved Application Compatibility and Deployment Support
- Dynamic Language Runtime
- Managed Extensibility Framework
- Parallel Programming framework
- Improved Security Model
- Networking Improvements
- Improved Core ASP.NET Services
- Improvements in WPF 4
- Improved Entity Framework (EF) Integration between WCF and WF

4. What is an IL?

Intermediate Language is also known as MSIL (Microsoft Intermediate Language) or CIL (Common Intermediate Language). All .NET source code is compiled to IL. IL is then converted to machine code at the point where the software is installed, or at run-time by a Just-In-Time (JIT) compiler.

5. What is Manifest?

Assembly metadata is stored in Manifest. Manifest contains all the metadata needed to do the following things

- Version of assembly.
- Security identity.
- Scope of the assembly.
- Resolve references to resources and classes.

The assembly manifest can be stored in a PE file either (an .exe or) .dll with Microsoft intermediate language (MSIL code with Microsoft intermediate language (MSIL) code or in a stand-alone PE file, that contains only assembly manifest information.

6. What are code contracts?

Code contracts help you to express the code assumptions and statements stating the behavior of your code in a language-neutral way. The contracts are included in the form of pre-conditions, post-conditions and objectinvariants. The contracts help you to improve-testing by enabling run-time checking, static contract verification, and documentation generation.

The *System.Diagnostics.Contracts* namespace contains static classes that are used to express contracts in your code.

7. Name the classes that are introduced in the *System.Numerics* namespace.

The following two new classes are introduced in the *System.Numerics* namespace:

- *BigInteger* - Refers to a non-primitive integral type, which is used to hold a value of any size. It has no lower and upper limit, making it possible for you to perform arithmetic calculations with very large numbers, even with the numbers which cannot hold by double or long.
- *Complex* - Represents complex numbers and enables different arithmetic operations with complex numbers. A number represented in the form *a + bi*, where *a* is the real part, and *b* is the imaginary part, is a complex number.

8. What is managed extensibility framework?

Managed extensibility framework (MEF) is a new library that is introduced as a part of .NET 4.0 and Silverlight 4. It helps in extending your application by providing greater reuse of applications and components. MEF provides a way for host application to consume external extensions without any configuration requirement.

9. Explain memory-mapped files.

Memory-mapped files (MMFs) allow you map the content of a file to the logical address of an application. These files enable the multiple processes running on the same machine to share data with each Other. The *MemoryMappedFile.CreateFromFile()* method is used to obtain a *MemoryMappedFile* object that represents a persisted memory-mapped file from a file on disk.

These files are included in the *System.IO.MemoryMappedFiles* namespace. This namespace contains four classes and three enumerations to help you access and secure your file mappings.

10. What is Common Type System (CTS)?

CTS is the component of CLR through which .NET Framework provides support for multiple languages because it contains a type system that is common across all the languages. Two CTS-compliant languages do not require type conversion when calling the code written in one language from within the code written in another language. CTS provide a base set of data types for all the languages supported by.NET Framework. This means that the size of integer and long variables is same across all .NET-compliant programming languages. However, each language uses aliases for the base data types provided by CTS. For example, CTS uses the data type system. int32 to represent a 4 byte integer value; however, Visual Basic uses the alias integer for the same; whereas, C# uses the alias int. This is done for the sake of clarity and simplicity.

11. Give a brief introduction on side-by-side execution. Can two applications, one using private assembly and the other using the shared assembly be stated as side-by-side executables?

Side-by-side execution enables you to run multiple versions of an application or component and CLR on the same computer at the same time. As versioning is applicable only to shared assemblies and not to private assemblies, two applications, one using a private assembly and other using a shared assembly, cannot be stated as side-by-side executables.

12. Which method do you use to enforce garbage collection in .NET?

The *System.GC.Collect()* method.

13. State the differences between the *Dispose()* and *Finalize()*.

CLR uses the Dispose and Finalize methods to perform garbage collection of run-time objects of .NET applications.

The *Finalize* method is called automatically by the runtime. CLR has a garbage collector (GC), which periodically checks for objects in heap that are no longer referenced by any object or program. It calls the Finalize method to free the memory used by such objects. The *Dispose* method is called by the programmer. *Dispose* is another method to release the memory used by an object. The Dispose method needs to be explicitly called in code to dereference an object from the heap. The *Dispose* method can be invoked only by the classes that implement the *IDisposable* interface.

14. What is code access security (CAS)?

Code access security (CAS) is part of the .NET security model that prevents unauthorized access of resources and operations, and restricts the code to perform particular tasks.

15. Differentiate between managed and unmanaged code?

Managed code is the code that is executed directly by the CLR instead of the operating system. The code compiler first compiles the managed code to intermediate language (IL) code, also called as MSIL code. This code doesn't depend on machine configurations and can be executed on different machines.

Unmanaged code is the code that is executed directly by the operating system outside the CLR environment. It is directly compiled to native machine code which depends on the machine configuration.
In the managed code, since the execution of the code is governed by CLR, the runtime provides different services, such as garbage collection, type checking, exception handling, and security support. These services help provide uniformity in platform and language-independent behavior of managed code applications. In the unmanaged code, the allocation of memory, type safety, and security is required to be taken care of by the developer. If the unmanaged code is not properly handled, it may result in memory leak. Examples of unmanaged code are ActiveX components and Win32 APIs that execute beyond the scope of native CLR.

16. What are tuples?

Tuple is a fixed-size collection that can have elements of either same or different data types. Similar to arrays, a user must have to specify the size of a tuple at the time of declaration. Tuples are allowed to hold up from 1 to 8 elements and if there are more than 8 elements, then the 8th element can be defined as another tuple. Tuples can be specified as parameter or return type of a method.

17. How can you turn-on and turn-off CAS?

YOU can use the Code Access Security Tool (Caspol.exe) to turn security on and off.

To turn off security, type the following command at the command prompt:
caspol -security off

To turn on security, type the following command at the command prompt:
caspol -security on

In the .NET Framework 4.0, for using Caspol.exe, you first need to set the *<LegacyCasPolicy>* element to *true*.

18. What is garbage collection? Explain the difference between garbage collections in .NET 4.0 and earlier versions.

Garbage collection prevents memory leaks during execution of programs. Garbage collector is a low-priority process that manages the allocation and deallocation of memory for your application. It checks for the unreferenced variables and objects. If GC finds any object that is no longer used by the application, it frees up the memory from that object.

GC has changed a bit with the introduction of .NET 4.0. In .NET 4.0, the *GC.Collect()* method contains the following overloaded methods:

```
GC.Collect(int)
GC.Collect(int, GCCollectionMode)
```

Another new feature introduced in .NET is to notify you when the *GC.Collect()* method is invoked and completed successfully by using different methods. The .NET 4.0 supports a new background garbage collection that replaces the concurrent garbage collection used in earlier versions. This concurrent GC allocates memory while running and uses current segment (which is 16 MB on a workstation) for that. After that, all threads are suspended. In case of background GC, a separate ephemeral GC - gen0 and gen1 can be started, while the full GC - gen0, 1, and 2 - is already running.

19. How does CAS works?

There are two key concepts of CAS security policy- code groups and permissions. A code group contains assemblies in it in a manner that each .NET assembly is related to a particular code group and some permissions are granted to each code group. For example, using the default security policy, a control downloaded from a Web site belongs to the Zone, Internet code group, which adheres to the permissions defined by the named permission set. (Normally, the named permission set represents a very restrictive range of permissions.)

Assembly execution involves the following steps:

1. Evidences are gathered about assembly.
2. Depending on the gathered evidences, the assembly is assigned to a code group.
3. Security rights are allocated to the assembly, depending on the code group.
4. Assembly runs as per the rights assigned to it.

20. What is Difference between NameSpace and Assembly?

Following are the differences between namespace and assembly:

- Assembly is physical grouping of logical units, Namespace, logically groups classes.
- Namespace can span multiple assembly.

21. Mention the execution process for managed code.

A piece of managed code is executed as follows:

- Choosing a language compiler
- Compiling the code to MSIL Compiling MSIL to native code Executing the code.

22. Is there a way to suppress the finalize process inside the garbage collector forcibly in .NET?

Use the *GC.SuppressFinalize()* method to suppress the finalize process inside the garbage collector forcibly in .NET.

23. How can you instantiate a tuple?

The following are two ways to instantiate a tuple:

- Using the *new* operator. For example,

```
Tuple<String, int> t = new Tuple<String, int> ("Hellow", 2);
```

- Using the *Create* factory method available in the Tuple class. For example,

```
Tuple<int, int, int> t = Tuple.Create<int, int, int> (2, 4, 5);
```

24. Which is the root namespace for fundamental types in .NET Framework?

System.Object is the root namespace for fundamental types in .NET Framework.

25. What are the improvements made in CAS in .NET 4.0?

The CAS mechanism in .NET is used to control and configure the ability of managed code. Earlier, as this policy was applicable for only native applications, the security guarantee was limited. Therefore, developers used to look for alternating solutions, such as operating system-level solutions. This problem was solved in .NET Framework 4 by turning off the machine-wide security. The shared and hosted Web applications can now run more securely. The security policy in .NET Framework 4 has been simplified using the transparency model. This model allows you to run the Web applications without concerning about the CAS policies.

As a result of security policy changes in .NET Framework 4.0, you may encounter compilation warnings and runtime exceptions, if your try to use the obsolete CAS policy types and members either implicitly or explicitly. However, you can avoid the warnings and errors by using the *<NetFx40_LegacySecurityPolicy>* configuration element in the runtime settings schema to opt into the obsolete CAS policy behavior.

26. What is Microsoft Intermediate Language (MSIL)?

The .NET Framework is shipped with compilers of all .NET programming languages to develop programs.

There are separate compilers for the Visual Basic, C#, and Visual C++ programming languages in .NET Framework. Each .NET compiler produces an intermediate code after compiling the source code. The intermediate code is common for all languages and is understandable only to .NET environment. This intermediate code is known as MSIL.

27. What is lazy initialization?

Lazy initialization is a process by which an object is not initialized until it is first called in your code. The .NET 4.0 introduces a new wrapper class, *System.Lazy<T>*, for executing the lazy initialization in your application. Lazy initialization helps you to reduce the wastage of resources and memory requirements to improve performance. It also supports thread-safety.

28. How many types of generations are there in a garbage collector?

Memory management in the CLR is divided into three generations that are build up by grouping memory segments. Generations enhance the garbage collection performance. The following are the three types of generations found in a garbage collector:

- Generation 0 - When an object is initialized, it is said to be in generation 0.
- Generation 1 - The objects that are under garbage collection process are considered to be in generation 1.
- Generation 2 - Whenever new objects are created and added to the memory, they are added to generation 0 and the old objects in generation 1 are considered to be in generation 2.

29. Explain covariance and contra-variance in .NET Framework 4.0. Give an example for each.

In .NET 4.0, the CLR supports covariance and contravariance of types in generic interfaces and delegates. Covariance enables you to cast a generic type to its base types, that is, you can assign a instance of type *IEnumerable<Tl>* to a variable of type *IEnumerable<T2>* where, *T1* derives from *T2*. For example,

```
IEnumerable<string> str1= new List<string> ();
IEnumerable<object> str2= str1;
```

Contravariance allows you to assign a variable of *Action<base>* to a variable of type *Action<derived>*. For example,

```
IComparer<object> obj1 = GetComparer()
IComparer<string> obj2 = obj1;
```

.NET framework 4.0 uses some language keywords (out and in) to annotate covariance and contra-variance. Out is used for covariance, while in is used for contra-variance.

Variance can be applied only to reference types, generic interfaces, and generic delegates. These cannot be applied to value types and generic types.

30. How do you instantiate a complex number?

The following are the different ways to assign a value to a complex number:

By passing two *Double* values to its constructor. The first value represents the real, and the second value represents imaginary part of a complex number. For example,

```
Complex c1 = new Complex(5, 8); /* It represents (5, 8) */
```

By assigning a *Byte*, *SByte*, *Intl6*, *UIntl6*, *Int32*, *UInt32*, *Int64*, *UInt64*, *Single*, or *Double* value to a *Complex* object. The assigned value represents the real part of the complex number, and its imaginary part becomes *0*. For example,

```
Complex c2 = 15.3; /* It represents (15.3, 0) */
```

By casting a *Decimal* or *BigInteger* value to a *Complex* object. For example,

```
Complex c3 = (Complex) 14.7; /* It represents (14.7, 0) */
```

Assigning the value returned by an operator to a *Complex* variable. For example,

```
Complex c4 = c1 + c2; /* It represents (20.3, 8) */
```

31. What is Common Language Specification (CLS)?

CLS is a set of basic rules, which must be followed by each .NET language to be a .NET- compliant language. It enables interoperability between two .NET-compliant languages. CLS is a subset of CTS; therefore, the languages supported by CLS can use each other's class libraries similar to their own. Application programming interfaces (APIs), which are designed by following the rules defined in CLS can be used by all .NET-compliant languages.

32. What is the role of the JIT compiler in .NET Framework?

The JIT compiler is an important element of CLR, which loads MSIL on target machines for execution. The MSIL is stored in .NET assemblies after the developer has compiled the code written in any .NET-compliant programming language, such as Visual Basic and C#.

JIT compiler translates the MSIL code of an assembly and uses the CPU architecture of the target machine to execute a .NET application. It also stores the resulting native code so that it is accessible for subsequent calls. If a code executing on a target machine calls a non-native method, the JIT compiler converts the MSIL of that method into native code. JIT compiler also enforces type-safety in runtime environment of .NET Framework. It checks for the values that are passed to parameters of any method.

For example, the JIT compiler detects any event, if a user tries to assign a 32-bit value to a parameter that can only accept 8-bit value.

33. What is difference between *System.String* and *System.StringBuilder* classes?

String and *StringBuilder* classes are used to store string values but the difference in them is that String is immutable (read only) by nature, because a value once assigned to a String object cannot be changed after its creation. When the value in the String object is modified, a new object is created, in memory, with a new value assigned to the *String* object. On the other hand, the *StringBuilder* class is mutable, as it occupies the same space even if you change the value. The *StringBuilder* class is more efficient where you have to perform a large amount of string manipulation.

34. Describe the roles of CLR in .NET Framework.

CLR provides an environment to execute .NET applications on target machines. CLR is also a common runtime environment for all .NET code irrespective of their programming language, as the compilers of respective language in .NET Framework convert every source code into a common language known as MSIL or IL (Intermediate Language).

CLR also provides various services to execute processes, such as memory management service and security services. CLR performs various tasks to manage the execution process of .NET applications.

The responsibilities of CLR are listed as follows:

- Automatic memory management
- Garbage Collection
- Code Access Security
- Code verification
- JIT compilation of .NET code

35. What is the difference between int and int32.

There is no difference between *int* and *int32*. *System.Int32* is a .NET Class and *int* is an alias name for *System.Int32*.

.NET Programming Concepts

1. Define variable and constant.

A variable can be defined as a meaningful name that is given to a data storage location in the computer memory that contains a value. Every variable associated with a data type determines what type of value can be stored in the variable, for example an integer, such as 100, a decimal, such as 30.05, or a character, such as 'A'.

You can declare variables by using the following syntax:

<Data_type> <variable_name> ;

A constant is similar to a variable except that the value, which you assign to a constant, cannot be changed, as in case of a variable. Constants must be initialized at the same time they are declared. You can declare constants by using the following syntax:

const int interestRate = 10;

2. What is a data type? How many types of data types are there in .NET ?

A data type is a data storage format that can contain a specific type or range of values. Whenever you declare variables, each variable must be assigned a specific data type. Some common data types include integers, floating point, characters, and strings. The following are the two types of data types available in .NET:

- **Value type -** Refers to the data type that contains the data. In other words, the exact value or the data is directly stored in this data type. It means that when you assign a value type variable to another variable, then it copies the value rather than copying the reference of that variable. When you create a value type

variable, a single space in memory is allocated to store the value (stack memory). Primitive data types, such as int, float, and char are examples of value type variables.

- **Reference type** - Refers to a data type that can access data by reference. Reference is a value or an address that accesses a particular data by address, which is stored elsewhere in memory (heap memory). You can say that reference is the physical address of data, where the data is stored in memory or in the storage device. Some built-in reference types variables in .Net are string, array, and object.

3. Mention the two major categories that distinctly classify the variables of C# programs.

Variables that are defined in a C# program belong to two major categories: **value type** and **reference type**. The variables that are based on value type contain a value that is either allocated on a stack or allocated in-line in a structure. The variables that are based on reference types store the memory address of a variable, which in turn stores the value and are allocated on the heap. The variables that are based on value types have their own copy of data and therefore operations done on one variable do not affect other variables. The reference-type variables reflect the changes made in the referring variables.

Predict the output of the following code segment:

```
int x = 42;
int y = 12;
int w;
object o; o
= x;
w = y * (int)o;  Console.WriteLine(w);

/* The output of the code is 504. */
```

4. Which statement is used to replace multiple if-else statements in code.

In Visual Basic, the **Select-Case** statement is used to replace multiple **If - Else** statements and in C#, the **switch-case** statement is used to replace multiple **if-else** statements.

5. What is the syntax to declare a namespace in .NET?

In .NET, the namespace keyword is used to declare a namespace in the code.

The syntax for declaring a namespace in C# is:
namespace UserNameSpace;

The syntax for declaring a namespace in VB is: *Namespace UserNameSpace*

6. What is the difference between constants and read-only variables that are used in programs?

Constants perform the same tasks as read-only variables with some differences. The differences between constants and read-only are

Constants:

1. Constants are dealt with at compile-time.
2. Constants supports value-type variables.
3. Constants should be used when it is very unlikely that the value will ever change.

Read-only:

1. Read-only variables are evaluated at runtime.
2. Read-only variables can hold reference type variables.
3. Read-only variables should be used when run-time calculation is required.

7. Differentiate between the *while* and *for* loop in C#.

The *while* and *for* loops are used to execute those units of code that need to be repeatedly executed, unless the result of the specified condition evaluates to false. The only difference between the two is in their syntax. The *for* loop is distinguished by setting an explicit loop variable.

8. What is an identifier?

Identifiers are northing but names given to various entities uniquely identified in a program. The name of identifiers must differ in spelling or casing. For example, *MyProg* and *myProg* are two different identifiers. Programming languages, such as C# and Visual Basic, strictly restrict the programmers from using any keyword as identifiers. Programmers cannot develop a class whose name is *public*, because, *public* is a keyword used to specify the accessibility of data in programs.

9. What does a break statement do in the switch statement?

The *switch* statement is a selection control statement that is used to handle multiple choices and transfer control to the *case* statements within its body. The following code snippet shows an example of the use of the *switch* statement in C#:

```
switch(choice)
{  case 1:
console.WriteLine("First");
break;  case 2:
```

```
console.WriteLine("Second");
break;  default:
 console.WriteLine("Wrong choice");
 break; }
```

In *switch* statements, the *break* statement is used at the end of a *case* statement. The *break* statement is mandatory in C# and it avoids the fall through of one *case* statement to another.

10. Explain keywords with example.

Keywords are those words that are reserved to be used for a specific task. These words cannot be used as identifiers. You cannot use a keyword to define the name of a variable or method. Keywords are used in programs to use the features of object-oriented programming.

For example, the *abstract* keyword is used to implement abstraction and the inherits keyword is used to implement inheritance by deriving subclasses in C# and Visual Basic, respectively.

The *new* keyword is universally used in C# and Visual Basic to implement encapsulation by creating objects.

11. Briefly explain the characteristics of value-type variables that are supported in the C# programming language.

The variables that are based on value types directly contain values. The characteristics of value-type variables that are supported in C# programming language are as follows:

- All value-type variables derive implicitly from the *System.ValueType* class
- You cannot derive any new type from a value type
- Value types have an implicit default constructor that initializes the default value of that type The value type consists of two main categories: Structs - Summarizes small groups of related variables.
    - Enumerations - Consists of a set of named constants.

12. Give the syntax of using the *while* loop in a C# program.

The syntax of using the *while* loop in C# is:

```
while(condition) //condition
{
 //statements
}
```

You can find an example of using the *while* loop in C#:

```
int i = 0; while(i
< 5)
{
 Console.WriteLine("{0} ", i);
i++; }
```

The output of the preceding code is: *0 1 2 3 4* .

13. What is a parameter? Explain the new types of parameters introduced in C# 4.0.

A parameter is a special kind of variable, which is used in a function to provide a piece of information or input to a caller function. These inputs are called arguments. In C#, the different types of parameters are as follows:

- **Value type** - Refers that you do not need to provide any keyword with a parameter.

- **Reference type** - Refers that you need to mention the *ref* keyword with a parameter.
- **Output type** - Refers that you need to mention the *out* keyword with a parameter.
- **Optional parameter** - Refers to the new parameter introduced in C# 4.0. It allows you to neglect the parameters that have some predefined default values. The example of optional parameter is as follows:
- `public int Sum(int a, int b, int c = 0, int d = 0); /* c and d is optional */`
- `Sum(10, 20); //10 + 20 + 0 + 0`
- `Sum(10, 20, 30); //10 + 20 + 30 + 0 Sum(10, 20, 30, 40); //10 + 20 + 30 + 40`
- **Named parameter** - Refers to the new parameter introduced in C# 4.0. Now you can provide arguments by name rather than position. The example of the named parameter is as follows:
- `public void CreateAccount(string name, string address = "unknown", int age = 0);`
- `CreateAccount("Sara", age: 30);`
- `CreateAccount(address: "India", name: "Sara");`

14. Briefly explain the characteristics of reference-type variables that are supported in the C# programming language.

The variables that are based on reference types store references to the actual data. The keywords that are used to declare reference types are:

1. **Class** - Refers to the primary building block for the programs, which is used to encapsulate variables and methods into a single unit.
2. **Interface** - Contains only the signatures of methods, properties, events, or indexers.
3. **Delegate** - Refers to a reference type that is used to encapsulate a named or anonymous method.

15. What are the different types of literals?

A literal is a textual representation of a particular value of a type.

The different types of literals in Visual Basic are:

- Boolean Literals - Refers to the True and False literals that map to the true and false state, respectively. Integer Literals - Refers to literals that can be decimal (base 10), hexadecimal (base 16), or octal (base 8).
- Floating-Point Literals - Refers to an integer literal followed by an optional decimal point By default, a floating-point literal is of type Double.
- String Literals - Refers to a sequence of zero or more Unicode characters beginning and ending with an ASCII double-quote character.
- Character Literals - Represents a single Unicode character of the Char type.
- Date Literals - Represents time expressed as a value of the Date type.
- Nothing - Refers to a literal that does not have a type and is convertible to all types in the type system.

The different types of literals in C# are:

- Boolean literals - Refers to the True and False literals that map to the true and false states, respectively.

- Integer literals - Refers to literals that are used to write values of types int, uint, long, and ulong.
- Real literals - Refers to literals that are used to write values of types float, double, and decimal.
- Character literals - Represents a single character that usually consists of a character in quotes, such as 'a'.
- String literals - Refers to string literals, which can be of two types in C#:
    - A regular string literal consists of zero or more characters enclosed in double quotes, such as "hello".
    - A verbatim string literal consists of the @ character followed by a double-quote character, such as @"hello".
- The Null literal - Represents the null-type.

16. What is the main difference between sub-procedure and function?

The sub-procedure is a block of multiple visual basic statements within Sub and End Sub statements. It is used to perform certain tasks, such as changing properties of objects, receiving or processing data, and displaying an output. You can define a sub-procedure anywhere in a program, such as in modules, structures, and classes.

We can also provide arguments in a sub-procedure; however, it does not return a new value.
The function is also a set of statements within the Function and End Function statements. It is similar to subprocedure and performs the same task. The main difference between a function and a sub-procedure is that subprocedures do not return a value while functions do.

17. Determine the output of the code snippet.

```
int a = 29;
a--; a -=
++a;
Console.WriteLine("The value of a is: {0}", a);

/* The output of the code is -1. */
```

18. Differentiate between Boxing and Unboxing.

When a value type is converted to an object type, the process is known as boxing; whereas, when an object type is converted to a value type, the process is known as unboxing.

Boxing and unboxing enable value types to be treated as objects. Boxing a value type packages it inside an instance of the Object reference type. This allows the value type to be stored on the garbage collected heap. Unboxing extracts the value type from the object. In this example, the integer variable *i* is boxed and assigned to object *obj*.

Example:

```
int i = 123;
object obj = i; /* Thi line boxes i. */
/* The object obj can then be unboxed and assigned to integer variable i: */ i
= (int)obj; // unboxing
```

19. Give the syntax of using the *for* loop in C# code?

The syntax of using the *for* loop in C# code is given as follows:

```
for(initializer; condition; loop expression) {
 //statements
}
```

In the preceding syntax, initializer is the initial value of the variable, condition is the expression that is checked before the execution of the *for* loop, and loop expression either increments or decrements the loop counter.

The example of using the *for* loop in C# is shown in the following code snippet:

```
for(int i = 0; i < 5; i++)
 Console.WriteLine("Hello");
```

In the preceding code snippet, the word Hello will be displayed for five times in the output window.

Object-Oriented Programming

1. What is object-oriented programming (OOP)?

OOP is a technique to develop logical modules, such as classes that contain properties, methods, fields, and events. An object is created in the program to represent a class. Therefore, an object encapsulates all the features, such as data and behavior that are associated to a class. OOP allows developers to develop modular programs and assemble them as software. Objects are used to access data and behaviors of different software modules, such as classes, namespaces, and sharable assemblies. .NET Framework supports only OOP languages, such as Visual Basic .NET, Visual C#, and Visual C++.

2. What is a class?

A class describes all the attributes of objects, as well as the methods that implement the behavior of member objects. It is a comprehensive data type, which represents a blue print of objects. It is a template of object.

A class can be defined as the primary building block of OOP. It also serves as a template that describes the properties, state, and behaviors common to a particular group of objects.

A class contains data and behavior of an entity. For example, the aircraft class can contain data, such as model number, category, and color and behavior, such as duration of flight, speed, and number of passengers. A class inherits the data members and behaviors of other classes by extending from them.

3. What is an object?

They are instance of classes. It is a basic unit of a system. An object is an entity that has attributes, behavior, and identity. Attributes and behavior of an object are defined by the class definition.

4. What is the relationship between a class and an object?

A class acts as a blue-print that defines the properties, states, and behaviors that are common to a number of objects. An object is an instance of the cass. For example, you have a class called *Vehicle* and *Car* is the object of that class. You can create any number of objects for the class named *Vehicle*, such as *Van*, *Truck*, and *Auto*.

The *new* operator is used to create an object of a class. When an object of a class is instantiated, the system allocates memory for every data member that is present in the class.

5. Explain the basic features of OOPs.

The following are the four basic features of OOP:

- **Abstraction** - Refers to the process of exposing only the relevant and essential data to the users without showing unnecessary information.
- **Polymorphism** - Allows you to use an entity in multiple forms.

- **Encapsulation** - Prevents the data from unwanted access by binding of code and data in a single unit called object.
- **Inheritance** - Promotes the reusability of code and eliminates the use of redundant code. It is the property through which a child class obtains all the features defined in its parent class. When a class inherits the common properties of another class, the class inheriting the properties is called a derived class and the class that allows inheritance of its common properties is called a base class.

6. What is the difference between arrays and collection?

Array:

1. You need to specify the size of an array at the time of its declaration. It cannot be resized dynamically.
2. The members of an array should be of the same data type.

Collection:

1. The size of a collection can be adjusted dynamically, as per the user's requirement. It does not have fixed size.
2. Collection can have elements of different types.
3. 7. What are collections and generics?
4. A collection can be defined as a group of related items that can be referred to as a single unit. The *System.Collections* namespace provides you with many classes and interfaces. Some of them are - *ArrayList*, *List*, *Stack*, *ICollection*, *IEnumerable*, and *IDictionary*. Generics provide the type-safety to your class at the compile time. While creating a data structure, you never need to specify the data type at the time of declaration. The *System.Collections.Generic* namespace contains all the generic collections.
5. 8. How can you prevent your class to be inherited further?
6. You can prevent a class from being inherited further by defining it with the *sealed* keyword.
7. 9. What is the index value of the first element in an array?
8. In an array, the index value of the first element is *0* (zero).
9. 10. Can you specify the accessibility modifier for methods inside the interface?
10. All the methods inside an interface are always *public*, by default. You cannot specify any other access modifier for them.
11. 11. Is it possible for a class to inherit the constructor of its base class?

12. No, a class cannot inherit the constructor of its base class.
13. 12. How is method overriding different from method overloading?
14. Overriding involves the creation of two or more methods with the same name and same signature in different classes (one of them should be parent class and other should be child).

    Overloading is a concept of using a method at different places with same name and different signatures within the same class.

13. What is the difference between a class and a structure?

Class:

1. A class is a reference type.
2. While instantiating a class, CLR allocates memory for its instance in heap.
3. Classes support inheritance.
4. Variables of a class can be assigned as null.
5. Class can contain constructor/destructor.

Structure:

1. A structure is a value type.
2. In structure, memory is allocated on stack.
3. Structures do not support inheritance.
4. Structure members cannot have null values.
5. Structure does not require constructor/destructor and members can be initialiazed automatically.

14. What are similarities between a class and a structure.

Structures and classes are the two most important data structures that are used by programmers to build modular programs by using OOP languages, such as Visual Basic .NET, and Visual C#. The following are some of the similarities between a class and a structure:

- Access specifiers, such as *public*, *private*, and *protected*, are identically used in structures and classes to restrict the access of their data and methods outside their body.
- The access level for class members and struct members, including nested classes and structs, is private by default. Private nested types are not accessible from outside the containing type.
- Both can have constructors, methods, properties, fields, constants, enumerations, events, and event handlers.
- Both structures and classes can implement interfaces to use multiple-inheritance in code.
- Both structures and classes can have constructors with parameter. Both structures and classes can have delegates and events.

15. What is a multicast delegate?

Each delegate object holds reference to a single method. However, it is possible for a delegate object to hold references of and invoke multiple methods. Such delegate objects are called multicast delegates or combinable delegates.

16. Can you declare an overridden method to be static if the original method is not static?

No. Two virtual methods must have the same signature.

17. Why is the virtual keyword used in code?

The *virtual* keyword is used while defining a class to specify that the methods and the properties of that class can be overridden in derived classes.

18. Can you allow a class to be inherited, but prevent a method from being overridden in C#?

Yes. Just declare the class *public* and make the method *sealed*.

19. Define enumeration?

Enumeration is defined as a value type that consists of a set of named values. These values are constants and are called enumerators. An enumeration type is declared using the *enum* keyword. Each enumerator in an enumeration is associated with an underlying type that is set, by default, on the enumerator. The following is an example that creates an enumeration to store different varieties of fruits:

enum Fruits {Mango, Apple, orange, Guava};

In the preceding example, an enumeration *Fruits* is created, where number *0* is associated with *Mango*, number *1* with *Apple*, number *2* with *Orange*, and number *3* with *Guava*. You can access the enumerators of an enumeration by these values.

20. In which namespace, all .NET collection classes are contained?

The *System.Collections* namespace contains all the collection classes.

21. Is it a good practice to handle exceptions in code?

Yes, you must handle exceptions in code so that you can deal with any unexpected situations that occur when a program is running. For example, dividing a number by zero or passing a string value to a variable that holds an integer value would result in an exception.

22. Explain the concept of constructor?

Constructor is a special method of a class, which is called automatically when the instance of a class is created. It is created with the same name as the class and initializes all class members, whenever you access the class. The main features of a constructor are as follows:

- Constructors do not have any return type
- Constructors are always public
- It is not mandatory to declare a constructor; it is invoked automatically by .NET Framework.

23. Can you inherit private members of a class?

No, you cannot inherit *private* members of a class because *private* members are accessible only to that class and not outside that class.

24. Does .NET support multiple inheritance?

.NET does not support multiple inheritance directly because in .NET, a class cannot inherit from more than one class. .NET supports multiple inheritance through interfaces.

25. How has exception handling changed in .NET Framework 4.0?

In .NET 4.0, a new namespace, *System.Runtime.ExceptionServices*, has been introduced which contains the following classes for handling exceptions in a better and advanced manner:

- *HandleProcessCorruptedStateExceptionsAttribute* Class - Enables managed code to handle the corrupted state exceptions that occur in an operating system. These exceptions cannot be caught by specifying the *try...catch* block. To handle such exceptions, you can apply this attribute to the method that is assigned to handle these exceptions.
- *FirstChanceExceptionEventArgs* Class - Generates an event whenever a managed exception first occurs in your code, before the common language runtime begins searching for event handlers.

26. What is a delegate?

A delegate is similar to a class that is used for storing the reference to a method and invoking that method at runtime, as required. A delegate can hold the reference of only those methods whose signatures are same as that of the delegate. Some of the examples of delegates are type-safe functions, pointers, or callbacks.

27. What is the syntax to inherit from a class in C#?

When a class is derived from another class, then the members of the base class become the members of the derived class. The access modifier used while accessing members of the base class specifies the access status of the base class members inside the derived class.

The syntax to inherit a class from another class in C# is as follows:
class MyNewClass : MyBaseclass

28. State the features of an interface.

An interface is a template that contains only the signature of methods. The signature of a method consists of the numbers of parameters, the type of parameter (value, reference, or output), and the order of parameters. An interface has no implementation on its own because it contains only the definition of methods without any method body. An interface is defined using the *interface* keyword. Moreover, you cannot instantiate an interface. The various features of an interface are as follows:

- An interface is used to implement multiple inheritance in code. This feature of an interface is quite different from that of abstract classes because a class cannot derive the features of more than one class but can easily implement multiple interfaces.
- It defines a specific set of methods and their arguments.
- Variables in interface must be declared as *public*, *static*, and *final* while methods must be *public* and *abstract*.
- A class implementing an interface must implement all of its methods. □ An interface can derive from more than one interface.

29. Can you use the 'throws' clause to raise an exception?

No, the *throws* clause cannot be used to raise an exception. The *throw* statement signals the occurrence of an exception during the execution of a program. When the program encounters a *throw* statement, the method terminates and returns the error to the calling method.

30. Define an array.

An array is defined as a homogeneous collection of elements, stored at contiguous memory locations, which can be referred by the same variable name. All the elements of an array variable can be accessed by index values. An Index value specifies the position of a particular element in an array variable.

31. What are methods?

Methods are the building blocks of a class, in which they are linked together to share and process data to produce the result. In other words, a method is a block of code that contains a series of statements and represents the behavior of a class. While declaring a method you need to specify the access specifier, the return value, the name of the method, and the method parameters. All these combined together is called the signature of the method.

32. What is a namespace?

Namespace is considered as a container that contains functionally related group of classes and other types.

33. Do events have return type?

No, events do not have return type.

34. What is the function of the Try-Catch-Finally block?

The *try* block encloses those statements that can cause exception and the *catch* block handles the exception, if it occurs. Catch block contains the statements that have to be executed, when an exception occurs. The *finally* block always executes, irrespective of the fact whether or not an exception has occurred. The *finally* block is generally used to perform the cleanup process. If any exception occurs in the *try* block, the program control directly transfers to its corresponding *catch* block and later to the *finally* block. If no exception occurs inside the *try* block, then the program control transfers directly to the *finally* block.

35. How can you prevent a class from overriding in C# and Visual Basic?

You can prevent a class from overriding in C# by using the *sealed* keyword; whereas, the *NotInheritable* keyword is used to prevent a class from overriding in Visual Basic.

36. What are abstract classes? What are the distinct characteristics of an abstract class?

An abstract class is a class that cannot be instantiated and is always used as a base class. The following are the characteristics of an abstract class:

- You cannot instantiate an abstract class directly. This implies that you cannot create an object of the abstract class; it must be inherited.
- You can have abstract as well as non-abstract members in an abstract class.
- You must declare at least one abstract method in the abstract class.
- An abstract class is always public.

- An abstract class is declared using the *abstract* keyword.

The basic purpose of an abstract class is to provide a common definition of the base class that multiple derived classes can share.

37. Give a brief description of properties in C# and the advantages that are obtained by using them in programs.

In C#, a property is a way to expose an internal data element of a class in a simple and intuitive manner. In other words, it is a simple extension of data fields. You can create a property by defining an externally available name and then writing the *set* and *get* property accessors. The get property accessor is used to return the property value. The set property accessor is used to assign a new value to the property.

38. Explain different types of inheritance.

Inheritance in OOP is of four types:

- **Single inheritance** - Contains one base class and one derived class
- **Hierarchical inheritance** - Contains one base class and multiple derived classes of the same base class
- **Multilevel inheritance** - Contains a class derived from a derived class
- **Multiple inheritance** - Contains several base classes and a derived class

All .NET languages supports single, hierarchical, and multilevel inheritance. They do not support multiple inheritance because in these languages, a derived class cannot have more than one base class. However, you can implement multiple inheritance in.NET through interfaces.

39. You have defined a destructor in a class that you have developed by using the C# programming language, but the destructor never executed. Why did the destructor not execute?

The runtime environment automatically invokes the destructor of a class to release the resources that are occupied by variables and methods of an object. However, in C#, programmers cannot control the timing for invoking destructors, as Garbage Collector is only responsible for releasing the resources used by an object.

Garbage Collector automatically gets information about unreferenced objects from .NET's runtime environment and then invokes the *Finalize()* method.

Although, it is not preferable to force Garbage Collector to perform garbage collection and retrieve all inaccessible memory, programmers can use the *Collect()* method of the Garbage Collector class to forcefully execute Garbage Collector.

40. What is a hashtable?

Hashtable is a data structure that implements the *IDictionary* interface. It is used to store multiple items and each of these items is associated with a unique string key. Each item can be accessed using the key associated with it. In short, hashtable is an object holding the key-value pairs.

41. Can users define their own exceptions in code?

Yes, customized exceptions can be defined in code by deriving from the *System.Exception* class.

42. Is it possible to execute two catch blocks?

You are allowed to include more than one *catch* block in your program; however, it is not possible to execute them in one go. Whenever, an exception occurs in your program, the correct *catch* block is executed and the control goes to the *finally* block.

43. What do you mean by data encapsulation?

Data encapsulation is a concept of binding data and code in single unit called object and hiding all the implementation details of a class from the user. It prevents unauthorized access of data and restricts the user to use the necessary data only.

44. What is the difference between procedural and object-oriented programming?

Procedural programming is based upon the modular approach in which the larger programs are broken into procedures. Each procedure is a set of instructions that are executed one after another. On the other hand, OOP is based upon objects. An object consists of various elements, such as methods and variables.

Access modifiers are not used in procedural programming, which implies that the entire data can be accessed freely anywhere in the program. In OOP, you can specify the scope of a particular data by using access modifiers - *public*, *private*, *internal*, *protected*, and *protected* internal.

45. Explain the concept of destructor?

A destructor is a special method for a class and is invoked automatically when an object is finally destroyed. The name of the destructor is also same as that of the class but is followed by a prefix tilde (~).

A destructor is used to free the dynamic allocated memory and release the resources. You can, however, implement a custom method that allows you to control object destruction by calling the destructor.

The main features of a destructor are as follows:

- Destructors do not have any return type
- Similar to constructors, destructors are also always public Destructors cannot be overloaded.

46. Can you declare a private class in a namespace?

The classes in a namespace are *internal*, by default. However, you can explicitly declare them as *public* only and not as *private*, *protected*, or *protected internal*. The nested classes can be declared as *private*, *protected*, or *protected internal*.

47. A structure in C# can implement one or more interfaces. Is it true or false?

Yes, it is true. Like classes, in C#, structures can implement one or more interfaces.

48. What is a static constructor?

Static constructors are introduced with C# to initialize the static data of a class. CLR calls the static constructor before the first instance is created.

The static constructor has the following features:

- No access specifier is required to define it.
- You cannot pass parameters in static constructor.
- A class can have only one static constructor.
- It can access only static members of the class.
- It is invoked only once, when the program execution begins.

49. What are the different ways a method can be overloaded?

The different ways to overload a method are given as follows:

- By changing the number of parameters used
- By changing the order of parameters
- By using different data types for the parameters 50. Differentiate between an abstract class and

an interface.

Abstract Class:

1. A class can extend only one abstract class
2. The members of abstract class can be private as well as protected. 3. Abstract classes should have subclasses
4. Any class can extend an abstract class.
5. Methods in abstract class can be abstract as well as concrete.
6. There can be a constructor for abstract class.
7. The class extending the abstract class may or may not implement any of its method.
8. An abstract class can implement methods.

Interface

1. A class can implement several interfaces 2.
An interface can only have public members.
3. Interfaces must have implementations by classes
4. Only an interface can extend another interface. 5. All methods in an interface should be abstract 6. Interface does not have constructor.
7. All methods of interface need to be implemented by a class implementing that interface.
8. Interfaces cannot contain body of any of its method.

51. What are queues and stacks?

Stacks refer to a list in which all items are accessed and processed on the Last-In-First-Out (LIFO) basis. In a stack, elements are inserted (push operation) and deleted (pop operation) from the same end called **top**.

Queues refer to a list in which insertion and deletion of an item is done on the First-In-First-Out (FIFO) basis. The items in a queue are inserted from the one end, called the **rear** end, and are deleted from the other end, called the **front** end of the queue.

52. Define an event.

Whenever an action takes place in a class, that class provides a notification to other classes or objects that are assigned to perform particular tasks. These notifications are called events. For example, when a button is clicked, the class generates an event called Click. An event can be declared with the help of the event keyword.

53. What are structures?

Structure is a heterogeneous collection of elements referenced by the same name. A structure is declared using the struct keyword. The following is an example that creates a structure to store an employee's information:

```
struct emp
{  fixed int
empID[15];  fixed char
name[30];  fixed char
addr[50];  fixed char
dept[15];  fixed char
desig[15];
}
```

The preceding example defines a structure *emp* and the members of this structure specify the information of an employee.

54. When do you really need to create an abstract class?

We define abstract classes when we define a template that needs to be followed by all the derived classes.

ADO.NET

1. What is the full form of ADO?

The full form of ADO is ActiveX Data Object.

2. Explain ADO.NET in brief.

ADO.NET is a very important feature of .NET Framework, which is used to work with data that is stored in structured data sources, such as databases and XML files. The following are some of the important features of ADO.NET:

- Contains a number of classes that provide you with various methods and attributes to manage the communication between your application and data source.
- Enables you to access different data sources, such as Microsoft SQL Server, and XML, as per your requirements.
- Provides a rich set of features, such as connection and commands that can be used to develop robust and highly efficient data services in .NET applications.
- Provides various data providers that are specific to databases produced by various vendors. For example, ADO.NET has a separate provider to access data from Oracle databases; whereas, another provider is used to access data from SQL databases.

3. What are major difference between classic ADO and ADO.NET?

Following are some major differences between both In ADO

we have recordset and in ADO.NET we have dataset.

- In recordset we can only have one table. If we want to accommodate more than one tables. We need to do inner join and fill the recordset. Dataset can have multiple tables.
- All data persist in XML as compared to classic ADO where data persisted in Binary format also.

4. What are the two fundamental objects in ADO.NET?

DataReader and *DataSet* are the two fundamental objects in ADO.NET.

5. What are the benefits of using of ADO.NET in .NET 4.0.

The following are the benefits of using ADO.NET in .NET 4.0 are as follows:

- **Language-Integrated Query (LINQ)** - Adds native data-querying capabilities to .NET languages by using a syntax similar to that of SQL. This means that LINQ simplifies querying by eliminating the need to use a separate query language. LINQ is an innovative technology that was introduced in .NET Framework 3.5.
- **LINQ to DataSet** - Allows you to implement LINQ queries for disconnected data stored in a dataset. LINQ to DataSet enables you to query data that is cached in a DataSet object. DataSet objects allow you to use a copy of the data stored in the tables of a database, without actually getting connected to the database.
- **LINQ to SQL** - Allows you to create queries for data stored in SQL server database in your .NET application. You can use the LINQ to SQL technology to translate a query into a SQL query and then use it to retrieve or manipulate data contained in tables of an SQL Server database. LINQ to SQL supports all the key functions that you like to perform while working with SQL, that is, you can insert, update, and delete information from a table.
- **SqlClient Support for SQL Server 2008** - Specifies that with the starting of .NET Framework version 3.5 Service Pack (SP) 1, .NET Framework Data Provider for SQL Server (*System.Data.SqlClient* namespace) includes all the new features that make it fully compatible with SQL Server 2008 Database Engine.
- **ADO.NET Data Platform** - Specifies that with the release of .NET Framework 3.5 Service Pack (SP) 1, an Entity Framework 3.5 was introduced that provides a set of Entity Data Model (EDM) functions. These functions are supported by all the data providers; thereby, reducing the amount of coding and maintenance in your application. In .NET Framework 4.0, many new functions, such as string, aggregate, mathematical, and date/time functions have been added.

6. Which namespaces are required to enable the use of databases in ASP.NET pages?

The following namespaces are required to enable the use of databases in ASP.NET pages:

- The *System.Data* namespace.
- The *System.Data.OleDb* namespace (to use any data provider, such as Access, Oracle, or SQL) The *System.Data.SQLClient* namespace (specifically to use SQL as the data provider)

7. Explain the *DataAdapter.Update()* and *DataSetAcceptChanges()* methods.

The *DataAdapter.Update()* method calls any of the DML statements, such as the *UPDATE*, *INSERT*, or *DELETE* statements, as the case may be to update, insert, or delete a row in a *DataSet*. The *DataSet.Acceptchanges()* method reflects all the changes made to the row since the last time the *AcceptChanges()* method was called.

8. What is the meaning of object pooling?

Object pooling is a concept of storing a pool (group) of objects in memory that can be reused later as needed. Whenever, a new object is required to create, an object from the pool can be allocated for this request; thereby, minimizing the object creation. A pool can also refer to a group of connections and threads. Pooling, therefore, helps in minimizing the use of system resources, improves system scalability, and performance.

9. Which properties are used to bind a *DataGridView* control?

The *DataSource* property and the *DataMember* property are used to bind a *DataGridView* control.

10. What property must be set and what method must be called in your code to bind the data from some data source to the Repeater control?

You must set the *DataSource* property and call the *DataBind()* method.

11. Mention the namespace that is used to include .NET Data Provider for SQL server in .NET code.

The *System.Data.SqlClient* namespace.

12. What is the difference between OLEDB Provider and SqlClient?

With respect to usage, there is no difference between OLEDB Provider and SqlClient. The difference lies in their performance. SqlClient is explicitly used to connect your application to SQL server directly, OLEDB Provider is generic for various databases, such as Oracle and Access including SQL Server.

Therefore, there will be an overhead which leads to performance degradation.

13. Name the two properties of the *GridView* control that have to be specified to turn on sorting and paging.

The properties of the *GridView* control that need to be specified to turn on sorting and paging are as follows:

- The *AllowSorting* property of the Gridview control indicates whether sorting is enabled or not. You should set the *AllowSorting* property to *True* to enable sorting.
- The *AllowPaging* property of the *Gridview* control indicates whether paging is enabled or not. You should set the *AllowPaging* property to *True* to enable paging.

14. Mention different types of data providers available in .NET Framework.

- .NET Framework Data Provider for SQL Server - Provides access to Microsoft SQL Server 7.0 or later version. It uses the *System.Data.SqlClient* namespace.
- .NET Framework Data Provider for OLE DB - Provides access to databases exposed by using OLE DB. It uses the *System.Data.OleDb* namespace.
- .NET Framework Data Provider for ODBC - Provides access to databases exposed by using ODBC. It uses the *System.Data.Odbc* namespace.

- .NET Framework Data Provider for Oracle - Provides access to Oracle database 8.1.7 or later versions. It uses the *System.Data.OracleClient* namespace.

15. Which architecture does Datasets follow?

Datasets follow the disconnected data architecture.

16. What is the role of the *DataSet* object in ADO.NET?

One of the major component of ADO.NET is the *DataSet* object, which always remains disconnected from the database and reduces the load on the database.

17. What is a *DataReader* object?

The *DataReader* object helps in retrieving the data from a database in a forward-only, read-only mode. The base class for all the *DataReader* objects is the *DbDataReader* class.

The *DataReader* object is returned as a result of calling the *ExecuteReader()* method of the *Command* object. The *DataReader* object enables faster retrieval of data from databases and enhances the performance of .NET applications by providing rapid data access speed. However, it is less preferred as compared to the *DataAdapter* object because the *DataReader* object needs an Open connection till it completes reading all the rows of the specified table.

An Open connection to read data from large tables consumes most of the system resources. When multiple client applications simultaneously access a database by using the *DataReader* object, the performance of data retrieval and other related processes is substantially reduced. In such a case, the database might refuse connections to other .NET applications until other clients free the resources.

18. How can you identify whether or not any changes are made to the *DataSet* object since it was last loaded?

The *DataSet* object provides the following two methods to track down the changes:

- The *GetChanges()* method - Returns the *DataSet* object, which is changed since it was loaded or since the *AcceptChanges()* method was executed.
- The *HasChanges()* method - Indicates if any changes occurred since the *DataSet* object was loaded or after a call to the *AcceptChanges()* method was made.

If you want to revert all changes since the *DataSet* object was loaded, use the *RejectChanges()* method.

19. Which property is used to check whether a *DataReader* is closed or opened?

The *IsClosed* property is used to check whether a *DataReader* is closed or opened. This property returns a *true* value if a Data Reader is closed, otherwise a *false* value is returned.

20. Name the method that needs to be invoked on the *DataAdapter* control to fill the generated *DataSet* with data?

The *Fill()* method is used to fill the dataset with data.

21. What is the use of the *Connection* object?

The *Connection* object is used to connect your application to a specific data source by providing the required authentication information in connection string. The connection object is used according to the type of the data source. For example, the *OleDbConnection* object is used with an OLE-DB provider and the *SqlConnection* object is used with an MS SQL Server.

22. What is the use of the *CommandBuilder* class?

The *CommandBuilder* class is used to automatically update a database according to the changes made in a *DataSet*.

This class automatically registers itself as an event listener to the *RowUpdating* event. Whenever data inside a row changes, the object of the *CommandBuilder* class automatically generates an SQL statement and uses the *SelectCommand* property to commit the changes made in *DataSet*.

OLEDB provider in .NET Framework has the *OleDbCommandBuiider* class; whereas, the SQL provider has the *SqlCommandBuilder* class.

23. Explain the architecture of ADO.NET in brief.

AD0.NET consists of two fundamental components:

- The *DataSet*, which is disconnected from the data source and does not need to know where the data that it holds is retrieved from.
- The .net data provider, which allows you to connect your application to the data source and execute the SQL commands against it.

The data provider contains the *Connection*, *Command*, *DataReader*, and *DataAdapter* objects. The *Connection* object provides connectivity to the database. The *Command* object provides access to database commands to retrieve and manipulate data in a database. The *DataReader* object retrieves data from the database in the readonly and forward-only mode. The *DataAdapter* object uses *Command* objects to execute SQL commands. The *DataAdapter* object loads the *DataSet* object with data and also updates changes that you have made to the data in the *DataSet* object back to the database.

24. Describe the disconnected architecture of ADO.NET's data access model.

ADO.NET maintains a disconnected database access model, which means, the application never remains connected constantly to the data source. Any changes and operations done on the data are saved in a local copy (dataset) that acts as a data source. Whenever, the connection to the server is re-established, these changes are sent back to the server, in which these changes are saved in the actual database or data source.

25. What are the usages of the *Command* object in ADO.NET?

The following are the usages of the *Command* object in AD0.NET:

The *Command* object in AD0.NET executes a command against the database and retrieves a *DataReader* or *DataSet* object.

- It also executes the *INSERT*, *UPDATE*, or *DELETE* command against the database.
- All the command objects are derived from the *DbCommand* class.
- The command object is represented by two classes: *SqlCommand* and *OleDbCommand*.
- The *Command* object provides three methods to execute commands on the database:
    - The *ExecuteNonQuery()* method executes the commands and does not return any value.
    - The *ExecuteScalar()* method returns a single value from a database query. ○ The *ExecuteReader()* method returns a result set by using the *DataReader* object.

26. What are the pre-requisites for connection pooling?

The prerequisites for connection pooling are as follows:

- There must be multiple processes to share the same connection describing the same parameters and security settings.
- The connection string must be identical.

27. What is connection pooling?

Connection pooling refers to the task of grouping database connections in cache to make them reusable because opening new connections every time to a database is a time-consuming process. Therefore, connection pooling enables you to reuse already existing and active database connections, whenever required, and increasing the performance of your application.

You can enable or disable connection pooling in your application by setting the pooling property to either true or false in connection string. By default, it is enabled in an application.

28. What are the various methods provided by the *DataSet* object to generate XML?

The various methods provided by the *DataSet* object to generate XML are:

- *ReadXml()* - Reads XML document into a *DataSet* object.
- *GetXml()* - Returns a string containing an XML document. *WriteXml()* - Writes an XML data to disk.

29. Out of Windows authentication and SQL Server authentication, which authentication technique is considered as a trusted authentication method?

The Windows authentication technique is considered as a trusted authentication method because the username and password are checked with the Windows credentials stored in the Active Directory.

The SQL Server Authentication technique is not trusted as all the values are verified by SQL Server only.

30. How would you connect to a database by using .NET?

The connection class is used to connect a .NET application with a database.

31. Which adapter should you use, if you want to get the data from an Access database?

OleDbDataAdapter is used to get the data from an Access database.

32. Which object is used to add a relationship between two *DataTable* objects?

The *DataRelation* object is used to add relationship between two *DataTable* objects.

33. What are different types of authentication techniques that are used in connection strings to connect .NET applications with Microsoft SQL Server?

.NET applications can use two different techniques to authenticate and connect with SQL Server. These techniques are as follows:

- The Windows Authentication option
- The SQL Server Authentication option

34. Explain the new features in ADO.NET Entity Framework 4.0.

ADO.NET Entity Framework 4.0 is introduced in .NET Framework 4.0 and includes the following new features:

- **Persistence Ignorance** - Facilitates you to define your own Plain Old CLR Objects (POCO) which are independent of any specific persistence technology.

- **Deferred or Lazy Loading** - Specifies that related entities can be loaded automatically whenever required. You can enable lazy loading in your application by setting the *DeferredLoadingEnabled* property to true.
- **Self-Tracking Entities** - Refers to the entities that are able to track their own changes. These changes can be passed across process boundaries and saved to the database.
- **Model-First Development** - Allows you to create your own EDM and then generate relational model (database) from that EDM with matching tables and relations.
- **Built-in Functions** - Enables you to use built-in SQL Server functions directly in your queries.
- **Model-Defined Functions** - Enables you to use the functions that are defined in conceptual schema definition language (CSDL).

35. What is the difference between the *Clone()* and *Copy()* methods of the *DataSet* class?

The *Clone()* method copies only the structure of a *DataSet*. The copied structure includes all the relation, constraint, and *DataTable* schemas used by the *DataSet*. The *Clone()* method does not copy the data, which is stored in the *DataSet*.

The *Copy()* method copies the structure as well as the data stored in the *DataSet*.

36. What is the use of *DataView*?

User-defined view of a table is contained in a *DataView*. A complete table or a small section of table depending on some criteria can be presented by an object of the *DataView* class. You can use this class to sort and find data within *DataTable*.

The *DataView* class has the following methods:

- *Find()* - Finds a row in a *DataView* by using sort key value.
- *FindRows()* - Uses the sort key value to match it with the columns of *DataRowView* objects. It returns an array of all the corresponding objects of *DataRowView* whose columns match with the sort key value.
- *AddNew()* - Adds a new row to the *DataView* object.
- *Delete()* - Deletes the specified row from the *DataView* object according to the specified index.

37. What are the parameters that control most of connection pooling behaviors?

The parameters that control most of connection pooling behaviors are as follows:

- Connect Timeout
- Max Pool Size
- Min Pool Size
- Pooling

38. How can you add or remove rows from the *DataTable* object of *DataSet*?

The *DataRowCollection* class defines the collection of rows for the *DataTable* object in a *DataSet*. The *DataTable* class provides the *NewRow()* method to add a new *DataRow* to *DataTable*. The *NewRow* method creates a new row, which implements the same schema as applied to the *DataTable*. The following are the methods provided by the *DataRowCollection* object:

- *Add()* - Adds a new row to *DataRowCollection*.
- *Remove()*- Removes a *DataRow* object from *DataRowCollection*.
- *RemoveAt()* - Removes a row whose location is specified by an index number.

39. Explain in brief DataAdapter class in ADO.NET.

The *DataAdapter* class retrieves data from the database, stores data in a dataset, and reflects the changes made in the dataset to the database. The *DataAdapter* class acts as an intermediary for all the communication between the database and the *DataSet* object. The *DataAdapter* Class is used to fill a *DataTable* or *DataSet* Object with data from the database using the *Fill()* method. The *DataAdapter* class applies the changes made in dataset to the database by calling the *Update()* method.

The *DataAdapter* class provides four properties that represent the database command:

SelectCommand, *InsertCommand*, *DeleteCommand*, and *UpdateCommand*.

Language-Integrated Query (LINQ)

1. What is Language Integrated Query (LINQ)?

LINQ is a programming model that is the composition of general-purpose standard query operators that allow you to work with data, regardless of the data source in any .NET based programming language. It is the name given to a set of technologies based on the integration of query capabilities into any .NET language.

2. What are LINQ query expressions?

A LINQ query, also known as a query expression, consists of a combination of query clauses that identify the data sources for the query. It includes instructions for sorting, filtering, grouping, or joining to apply to the source data. The LINQ query expressions syntax is similar to the SQL syntax. It specifies what information should be retrieved from the data source.

3. Write the basic steps to execute a LINQ query.

The following are the three basic steps to execute a LINQ query:

- Obtain the data source (The data source can be either an SQL database or an XML file)
- Create a query
- Execute the query

4. Write the basic syntax of a LINQ query in Visual Basic as well as in C#.

In Visual Basic, the basic syntax of a LINQ query starts with the **From** clause and ends with the **Select** or **Group By** clause. In addition, you can use the **Where**, **Order By**, and **Order By Descending** clauses to perform additional functions, such as filtering data and generating the data in a specific order.

In C#, the basic syntax of a LINQ query starts with the **From** clause and ends with the **Select** or **group by clause**. In addition, you can use the **where**, **orderby**, and **Orderby descending** clauses to perform additional functions, such as filtering data and generating the data in a specific order.

5. In which statement the LINQ query is executed?

A LINQ query is executed in the **For Each** statement in Visual Basic and in the **foreach** statement in C#.

6. In LINQ, lambda expressions underlie many of the standard query operators. Is it True or False?

It is true.

7. What is PLINQ?

PLINQ stands for Parallel Language Integrated Query. It is the parallel implementation of LINQ, in which a query can be executed by using multiple processors. PLINQ ensures the scalability of software on parallel processors in the execution environment. It is used where data grows rapidly, such as in telecom industry or where data is heterogeneous.

PLINQ also supports all the operators of LINQ. In addition, you can query 'collections by using PLINQ. It can also run several LINQ queries simultaneously and makes use of the processors on the system. Apart from this, PLINQ uses parallel execution, which helps in running the queries quickly. Parallel execution provides a major performance improvement to PLINQ over certain types of legacy code, which takes too much time to execute.

8. What are the different Visual Basic features that support LINQ?

Visual Basic includes the following features that support LINQ:

- **Anonymous types** - Enables you to create a new type based on a query result.

- **Implicitly typed variables** - Enables the compiler to infer and assign a type when you declare and initialize a variable.
- **Extension method** - Enables you to extend an existing type with your own methods without modifying the type itself.

9. What is the function of the DISTINCT clause in a LINQ query?

The **DISTINCT** clause returns the result set without the duplicate values.

10. What is the DataContext class and how is it related to LINQ?

After you add a LINQ to SQL Classes item to a project and open the O/R Designer, the empty design surface represents an empty *DataContext* class ready to be configured. The *DataContext* class is a LINQ to SQL class that acts as a conduit between a SQL Server database and the LINQ to SQL entity classes mapped to that database. This class contains the connection string information and the methods for connecting to a database and manipulating the data in the database. It is configured with connection information provided by the first item that is dragged onto the design surface.

11. What is the difference between the Take and Skip clauses?

The **Take** clause returns a specified number of elements. For example, you can use the **Take** clause to return two values from an array of numbers. The **Skip** clause skips the specified number of elements in the query and returns the rest. For example, you can use the **Skip** clause to skip the first four strings in an array of strings and return the remaining array of string.

12. What is Object Relational Designer (0/R Designer)?

The 0/R Designer provides a visual design surface to create LINQ to SQL entity classes and associations (relationships) that are based on objects in a database.

13. Which interface implements the standard query operators in LINQ?

The standard query operators implement the *IEnumerable<T>* or the *IQueryable<T>* interface in C# and the *IEnumerable(Of T)* or the *IQueryable(Of T)* interface in Visual Basic.

14. What are standard query operators in LINQ?

The standard query operators in LINQ are the extension methods that form the LINQ pattern. These operators form an API that enables querying of any .NET array or collection. It operates on sequences and allows you to perform operations, such as determining if a value exists in the sequence and performing an aggregated function, such as a summation over a sequence.

15. On what parameter does the *GroupBy* clause group the data?

The **GroupBy** clause groups the elements that share a common attribute.

16. What is a *LinqDataSource* control?

The *LinqDataSource* control enables you to use LINQ. in an ASP.NET Web page by setting the properties in the markup text. You can use the control retrieve or modify data. It is similar to the *SqIDataSource* and

ObjectDataSource controls in the sense that it can be used to declaratively bind other ASP.NET controls on a page to a data source. The difference is that instead of binding directly to a database or to a generic class, the *LinqDataSource* control is designed to bind a LINQ enabled data model.

17. How can you open the O/R Designer?

You can open the O/R Designer by adding a new LINQ to SQL Classes item to a project.

18. The standard query operators are themselves a set of extension methods that provide the LINQ query functionality for any type that implements the *IEnumerable<T>* interface in Visual Basic. Is it True or False?

False, as it implements the *IEnumerable(T)* interface in Visual Basic and the *IEnumerable<T>* interface is implemented in C#.

19. What are lambda expressions in LINQ?

A lambda expression is a function without a name that calculates and returns a single value. All lambda expressions use the lambda operator =>, which read as goes to. The left side of the lambda operator specifies the input parameters and the right side holds the expression or statement block.

20. Before you query a DataSet object by using LINQ to DataSet, you must first populate the dataset How can you do this?

You can load the data into the dataset by using different methods, such as:

- Using the *DataAdapter* class
- Using LINQ to SQL

21. What are the different implementations of LINQ?

The different implementations of LINQ are:

- **LINQ to SQL** - Refers to a component of.NET Framework version 3.5 that provides a run-time infrastructure to manage relational data as objects.
- **LINQ to DataSet** - Refers to a component that makes it easier and faster to query over data cached in a DataSet object.
- **LINQ to XML** - Provides an in-memory XML programming interface.

- **LINQ to Objects** - Refers to the use of LINQ queries with any *IEnumerable* or *IEnumerable(T)* collection directly, without the use of an intermediate LINQ provider or API, such as LINQ to SQL or LINQ to XML.

22. Which command-line tool generates code and mapping for the LINQ to SQL component of .NET Framework?

The **SqlMetal.exe** command-line tool generates code and map the LINQ to SQL component.

23. Name the control that exposes the LINQ features to Web developers through the ASP.NET data-source control architecture.

The *LinqDataSource* control exposes the LINQ features to Web developers through the ASP.NET data-source control architecture.

24. What is the difference between the Select clause and *SelectMany()* method in LINQ?

Both the Select clause and *SelectMany()* method are used to produce a result value from a source of values. The difference lies in the result set. The Select clause is used to produce one result value for every source value. The result value is a collection that has the same number of elements from the query. In contrast, the *SelectMany()* method produces a single result that contains a concatenated collection from the query.

25. Which extension method do you need to run a parallel query in PLINQ?

The **AsParallel** extension method is required to run a parallel query in PLINQ.

Dynamic Programming

1. What is Dynamic Language Runtime (DLR)?

DLR is a runtime environment that allows you to integrate dynamic languages with the Common Language Runtime (CLR) by adding a set of services, such as expression trees, call site caching, and dynamic object interoperability to the CLR.

The *System.Dynamic* and *System.Runtime.CompilerServices* namespaces are used to hold the classes for DLR. It also provides dynamic features to statically-typed languages, such as C# and Visual Basic to enable their interoperation with dynamic languages.

2. What are the advantages of DLR?

The various advantages provided by DLR are:

- Allows you to easily implement the dynamic languages to the .NET Framework.
- Provides dynamic features to statically-typed languages. The statically-typed .NET Framework languages, such as C# and Visual Basic can create dynamic objects and use them together with statically-typed objects.
- Implements sharing of libraries and objects, which means that the objects and libraries implemented in one language can be used by other languages using DLR. The DLR also enables interoperation between statically-typed and dynamic languages.
- Enables fast execution of dynamic operations by supporting advance caching.

3. Give a brief introduction to Binders.

Binders are used by DLR to communicate with not the .NET Framework but also with various other services, such as Silverlight and COM. These services represent language-specific semantics and specify how a particular operation can be performed at the call site.

Call sites refer to the area in the code where logical and mathematical operations, such as **a + b** or **a.b()** are performed on dynamic objects.

4. Explain the different services provided by DLR to CLR.

The services provided by DLR to CLR are used for supporting dynamic languages. These services include the following:

- **Expression Trees** - Refers to the representation of code in a data structure similar to a tree. However, expression trees in DLR are the advanced version of the expression trees that were introduced with LINQ in .NET 3.5. Therefore, DLR has extended the functionalities of Language Integrated Query (LINQ) expression trees, such as control flow, assignment, and other language-modeling nodes to a dynamic language. These expression trees define the semantics of a language in form of an **abstract syntax tree (AST)**. AST enables the DLR to dynamically generate code, which the CLR executes at runtime.
- **Call Site Caching** - Enables the DLR to store the information of the operations and characteristics of the variables, such as their data type. The call site caching services also enables to check whether such operations have been performed previously to retrieve all the information about the variable. The place where DLR stores these values is called a **call site**.
- **Dynamic Object Interoperability** - Enables the DLR to provide a set of classes and interfaces that represent dynamic objects and operations. These classes and interfaces can be used to create classes for dynamic libraries, which can be used in static and dynamic type languages.

5. Name the binders provided by .NET Framework 4.0.

.NET Framework 4.0 provides the following binders:

- **Object Binder** - Enables to communicate with .NET objects.
- **JavaScript Binder** - Enables to communicate with JavaScript in Silverlight.
- **Python Binder** - Enables to communicate with IronPython.
- **Ruby Binder** - Enables to communicate with IronRuby. **COM Binder** - Enables to communicate with COM.

6. Explain *ExpandoObject* and *DynamicObject* classes.

The *ExpandoObject* class refers to a class whose members can be explicitly added and removed at runtime. In other words, the *ExpandoObject* class allows dynamic binding of the objects, which enables you to use standard syntax, similar to the *dynobj.Method* method instead of using more complex syntax, such as *dynobj.getAttribute("Method")*.

The *DynamicObject* class enables you to define the dynamic behavior for an object at run time. This class cannot be instantiated directly; therefore, to implement the dynamic behavior, you must inherit from the *DynamicObject* class and override the necessary methods. It allows you to define the specific operations that can be performed on dynamic objects as well the methods to perform those operations.

7. What is the difference between dynamic and var data types?

The difference between the var and dynamic data types is that the var data type is strongly type checked at the compile time; whereas, the dynamic data type is type checked by the compiler only at run time. After declaring a var data type, you cannot explicitly change its type throughout the execution of the program; however, a variable of the dynamic data type can be changed during runtime. Another major difference between the two is that dynamic type can also be used as the return type for methods, for which var cannot be used.

8. Which class is used for converting the data types?

The *System.Convert* class provides a complete set of methods for converting the data types.

XML

1. What is Extensible Markup Language (XML).

XML is a simple and flexible markup language in the text format. Nowadays, it is widely used to exchange a large variety of data over the Internet. XML consists of data as text in well-defined customized layouts by using self-defining tags. These user-defined tags are user friendly because they contain the name given by the user and make the information easily understandable to a user. These user-friendly features made XML to be widely used as a standard data-interchange format. The World Wide Web Consortium (W3C) frequently develops new standard for XML usage by different software vendors and solution providers. XML plays a very significant role with respect to .NET Framework 4.0. .NET Framework 4.0 provides us with a namespace called *System.Xml*, which includes classes that are used to work with XML.

2. What is the version information in XML?

"Version" tag shows which version of XML is used.

3. If XML does not have closing tag will it work?

No, every tag in XML, which is opened, should have a closing tag.

4. Is XML case sensitive?

Yes, XML is case sensitive.

5. Explain the difference between XML and HTML. XML describes data while HTML describes how the data should be displayed. Therefore, HTML is about displaying information while XML is about describing information.
   - XML supports user-defined tags while HTML provides pre-defined tags.
   - XML is a case-sensitive language while HTML language is not case-sensitive.
   - In XML, all tags must be closed; while in HTML, it is not necessary to close each tag.

6. What is XML DOM?

The DOM stands for Document Object Model, which describes the logical formation of documents and provides the way to access and manipulate a document. It supplies an Application Programming Interface (API) to XML documents. It is built around the object-oriented design; therefore, it is known as DOM. The DOM model considers an XML document as a composition of objects and every object consists of properties and behaviors that can be manipulated by the DOM methods. The DOM allows creating and building XML documents, navigating the structure of documents, and managing the elements and their data. You can use the DOM methods and objects with any language, such as C#, VB, JavaScript, and VBScript.

7. Which namespaces in .NET are used for XML?

The *System.xml.dll* is the real physical file, which contains the XML implementation. Some of the other namespaces that allow .NET to use XML are as follows:

- *System.Xml*
- *System.Xml.Schema*
- *System.Xml.XPath*
- *System.Xml.Xsl*

8. Explain different types of XML Application Programming Interface (API).

The following are two main types of XML parsers:

- Tree-based API - Compiles an XML document into a tree structure and loads it into memory. You can traverse and change the tree structure. The DOM is an example of a tree-based API.
- Event-based API - Provides the report to an application about the parsing events by a set of built-in callback functions. An example of the event-based API is SAX.

9. Explain the *XmlReader* class.

The *XmlReader* class is used to read XML data in a fast, forward-only, and non-cached manner. To work with XmlReader class in .NET, you need to import the following namespace:

In C#:
using System.Xml;

In VB:
Imports System.Xml

10. Describe the *XmlWriter* class.

The *XmlWriter* class is used to write XML to a stream, a file, or a Textwriter object. This class works in a forward-only, non-cached manner. You can configure the *XmlWriter* object up to a large extent. With this object, you can specify a few things, such as whether to indent content or not, the amount to indent, what quote character to use in attribute values, and whether or not namespaces are supported.

11. What is XPath?

XPath stands for XML Path. It is a language used to access different parts of an XML document, such as elements and attributes.

12. What is an XML attribute?

An XML attribute contains additional information regarding that particular element. The XML attributes use the name-value pair. For example, the element student has an attribute called id and the value of this attribute is set to s01, as shown in the following code snippet:

<Student ID="s01">
...
</Student>

13. The XML elements cannot be empty. Is it true?

No, it is not true.

14. Describe the role that XSL can play while dynamically generating HTML pages from a relational database.

The SQLXML 3.0 and advanced versions provide the facility of mapping the SQL queries output with XSLT templates. It uses XSLT to present the records that are retrieved from databases on Web pages (HTML pages).

An application can use XSLT to modify the output that is retrieved from data sources and display the output by XSL templates. The XSLT displays data without affecting the database query and the code of application.

15. What are the advantages of DOM?

The following are the advantages of DOM:

- DOM stores the entire XML document into memory before processing. Therefore, the XML structure can be easily modified and values can be added, changed, and removed.
- DOM enables to traverse the XML structure in any direction. It means that you can access any node of the XML structure by traversing through the XML structure.

16. Give an example of a DOM-enabled XML parser.

The XML parser is MSXML, which is fully DOM-enabled.

17. What is an XML schema?

An XML schema provides the definition of an XML document. This implies that an XML schema defines the following in an XML document:

- The elements that can appear in an XML document.
- The attributes that can appear in an XML document.
- The elements that are child elements.
- The order of child elements.
- The number of child elements.
- Whether an element is empty or it includes some text. The data types for elements and attributes.

18. State the advantages of XML schemas over DTD.

Microsoft developed a language known as the XML Schema Definition (XSD) to describe the schema to an XML document. The following are the advantages of XML schemas over DTDs: XSD keeps much better control over types of data than the DTD.

- DTD does not allow creating customized data types while the XSD provides full support to create customized data types.
- XSD allows you to specify restrictions on data. It means that you can define the type of data that should be stored in an element, for example numbers or alphabets.
- The XSD is quite easy to learn and to understand because its syntax is same as that of the XML document.

19. Using XSLT, how would you extract the value of a specific attribute from an element in an XML document?

The components necessary for the above mentioned operation are as follows:

- The template element - Matches the correct XML element.
- The value-of element - Selects the attribute value.

- The optional apply-templates element - Allows continuous processing of the document

20. Which classes are supported to make an XML DOM?

The following are the different classes in the *System.Xml* namespace that make up the XML DOM:

- The *XmlNode* class
- The *XmlDocument* Class
- The *XmlElement* Class
- The *XmlAttribute* Class
- The *XmlText* class
- The *XmlComment* class
- The *XmlNodeList* Class

21. Which class is used to encode and decode XML names and contains different methods to convert between CLR types and XSD types.

The *XmlConvert* Class.

22. What is the DTD?

The DTD is Document Type Definition that describes the formation of the content of an XML document. The DTD manages the data to store in a consistent format. It defines the XML elements and attributes about how they should be present in XML documents and what relation they should have with other elements and attributes. The DTD also allows you to mention whether an XML element is optional or not. If the XML documents are not according to the DTD rules, they are not considered valid.

23. Is it true that the XML's goal is to replace HTML?

No, it is not true. Both are necessary in their respective fields.

24. What is XSLT?

XSLT is Extensible Stylesheet Language Transformations that is a part of XML, which is a mechanism to transform an XML document into another XML or HTML document.

25. Describe the rules and regulations that must be followed while creating a well-formed XML document.

The following are the rules and regulations that are necessary to follow while creating a well-formed XML document:

- Every start tag must end with an end tag.
- A root element should be included for enclosing other child elements.
- XML tags are case-sensitive; therefore, start and end tags must be of same spelling and the casing should also be the same.
- XML's empty tags are necessary to close with a forward slash (/).
- XML's attributes values are necessary to enclose within double quotation marks.

- XML tags must be properly nested. It means starting tags should be closed in the reverse order in which they present.

26. What are the naming conventions required for XML elements tags?

The following are the naming conventions that need to be followed for XML elements tags:

- Element names should contain only characters, numbers, hyphens, and periods.
- Element names cannot not begin with a number or punctuation character.
- Element names must not start with the word xml (or XML, or Xml).
- Element names cannot consist spaces.
- Element names can be used any words except xml, XML, or Xml because no words are reserved in XML.

27. The XML preserves white spaces. Is it true?

Yes, it is true.

28. Explain the XML elements.

The elements are the central units of an XML document that explain and identify data. The elements are represented by the tags. You can also make your own tags, which make XML a user-friendly language. By creating custom meaningful elements, you can improve readability of the document. XML elements can be nested and the nested elements are known as child elements.

Application Deployment

1. What is deployment?

Deployment refers to the distribution of an application among various end-users. It is a process that makes software available for use by just installing it on the client computer.

2. List different ways of deployment that are supported by .NET Framework 4.0.

- Windows Installer
- ClickOnce XCOPY
- Copy Web Site Publish Web Site tool

3. What is XCOPY?

XCOPY enables you to deploy an application by copying the application directory and all subdirectories to the target computer and then executing the application on the client. The application starts executing on the target computer by using its assembly file, which is a self-description file that contains all the information about the application. The XCOPY deployment does not make any impact on the target system while configuring the components and registering entries, and is therefore known as zero-impact installation.

4. Does *XCOPY* copy the hidden and system files?

No. By default, *XCOPY* excludes the hidden and system files. However, you can include the hidden and system files using the */h* switch.

5. Why do you use Windows Installer?

The Windows Installer deployment technique allows you to deploy Windows-based and Web applications by creating a Windows Installer Package. The installer package has an extension of *.msi* and it contains the application, any dependent files, registry entries, and the rest. The installer package can then be distributed to various end-users by simply copying it on the target computers.

The end-users can then run the installer package to install the application anywhere in their computers. The installation takes place using the installation wizard; therefore, the users can easily install the application on their system. Once your application is installed on the target computer, end-users can open the application from the installed location.

6. Can you deploy an ASP.NET Web application project using the Copy Web Site option?

No. The Copy Web Site option can only be used to deploy the Web sites.

7. How can you determine whether you should deploy the application or publish the application?

If you want to host the application on a shared hosting environment, you should use publishing; whereas, if you want to create a Web application that is downloaded from a Web site, you should deploy the application to create a setup.exe file.

8. How can you deploy an ASP.NET Web application?

You can deploy an ASP.NET Web application using either the Windows Installer deployment or ClickOnce deployment technique.

9. What is Application Cache?

When a ClickOnce application is installed locally or hosted online, it is stored in the ClickOnce application cache of the client computer. The ClickOnce application cache is a set of hidden directories placed under the Local Settings directory of the current user's Documents and Settings folder. The application cache contains all the application files, assemblies, configuration files, application and user settings, and data directory. In case the ClickOnce applications are hosted online, the size of the ClickOnce application cache gets limited to a specified amount; whereas, the installed applications do not restrict to the cache size limitation. The cache storage quota is responsible to determine the size of the application cache.

10. What are the enhancements in ClickOnce deployment in .NET 4.0?

In .NET 4.0, the ClickOnce deployment technology is enhanced with the following features:

- **Support for .NET Framework 4.0 version** - Creates applications by using Visual Studio 2010 that can target .NET Framework 4.0 and its new features.
- **Support for multiple versions of the .NET Framework** - Creates applications that are compatible with multiple versions of the .NET Framework. You can specify the target framework for an application as .NET Framework 3.5 or .NET Framework 4 while creating the application.

- **Enhanced logging feature** - Stores logging information that includes various parameters passed to the ClickOnce runtime, the browser settings, and ClickOnce security options.
- **Custom Installer and User Interface** - Allows you to create a custom graphical user interface for installing and updating the .exe applications. In addition, the custom installer can have custom dialog boxes for security and maintenance operations.

11. What is the difference between deploying and publishing an application?

In deployment, you can create a new setup and deployment project. In this project, you can add the project output and create a setup.exe file. After creating an executable file, you need to login into the server and execute the setup.exe file to install the application. On the other hand, in publishing, you need to right-click the application in the Solution Explorer and select Publish to publish the application. Then, you specify a location where the application is to be published. The users can then install the application from the location where you have published it and run locally even when the computer is offline.

12. What do you mean by Merge Module projects?

Merge Module projects are used to package the files and components that are shared between multiple applications. The Merge Module project file contains the *.msm* extension. The *.msm* file includes files, resources, registry entries, and setup logic. This file is merged with a Windows installer (*.msi*) file to correctly install the shared files. If a single merge module is used by more than one application, then you need to add that merge module in the package only once.

13. What is the need of Copy Web Site?

Copy Web Site is a tool used to deploy the Web site by copying its content files. The Copy Web Site tool also checks whether or not the latest version of a file is present at the destination. If files of the most recent version are found at the destination, then the Copy Web Site tool does not superimpose the older version of files. The Copy Web Site deployment tool consists of the following main entities:

- **Project source** - Specifies the source directory, which contains the contents and references of a Web site at development time. In simple words, you can say that the project source specifies the site that you currently have opened in Visual Studio 2010. The Copy Web Site tool picks all the files for deployment from this location.
- **Project destination** - Specifies the destination folder where you have to deploy the application. This destination directory can be placed on remote computers or servers, which allow you to copy the Web site contents using the Front Page Server Extensions, FTP, or HTTP protocol implementations for content transfer.
- **Synchronizing two Web sites** - Synchronizes two Web sites by copying each other's files. Synchronization checks the files on the local and remote sites and ensures that all files on both sites are up to date.

14. What is the use of the Copy Project command?

The Copy Project command copies only the files required to run the project and pastes it on the target server. It does not deploy the complete project; therefore, IIS directory settings are not automatically configured.

15. Can Windows applications and the Web applications be deployed using the same template of Setup and Deployment project?

No. the Windows applications use the Setup Project template; whereas, the Web applications use the Web Setup Project template. After the deployment, their installation takes place in the similar way.

16. Explain the .NET Framework deployment features.

In a general context, .NET Framework includes the following deployment features:

- **No-impact applications** - Provides application isolation and removes DLL conflicts.
- **Private components by default** - Enables the components to deploy to the application directory and to be visible only to the containing application.

- **Side-by-side versioning** - Enables you to select one of the multiple versions.
- **XCOPY deployment and replication** - Refers to the self-descriptive application that is deployed without the need to store registry entries.
- **On-the-fly updates** - Allows for the updating of the DLLs of the remote computers.
- **Integration with the Microsoft Windows Installer** - Makes the features, such as advertising, publishing, repairing, and install-on-demand available during deployment of an application.
- **Enterprise deployment** - Eases the task of software distribution.
- **Downloading and caching** - Specifies that the downloads are kept smaller and the components are isolated for application use.
- **Partially trusted code** - Enables code-based identification.

.NET Assemblies

1. What is an assembly?

Assemblies are the basic building blocks required for any application to function in the .NET realm. They are partially compiled code libraries that form the fundamental unit of deployment, versioning, activation scoping, reuse, and security. Typically, assemblies provide a collection of types and resources that work together to form a logical unit of functionality. They are the smallest deployable units of code in .NET. Compared to the executable files assemblies are far more reliable, more secure, and easy to manage. An assembly contains a lot more than the Microsoft Intermediate Language (MSIL) code that is compiled and run by the Common Language Runtime (CLR). In other words, you can say that an assembly is a set of one or more modules and classes compiled in MSIL, and metadata that describes the assembly itself, as well as the functionalities of the assembly classes.

2. Name the different components of an assembly.

An assembly is a logical unit that is made up of the following four different types of components:

- Assembly manifest
- MSIL source code
- Type metadata
- Resources

3. What are the different types of assemblies? Explain them in detail.

The following are the two types of assemblies:

- **Private Assembly** - Refers to the assembly that is used by a single application. Private assemblies are kept in a local folder in which the client application has been installed.
- **Public or Shared Assembly** - Refers to the assembly that is allowed to be shared by multiple applications. A shared assembly must reside in Global Assembly Cache (GAC) with a strong name assigned to it.

For example, imagine that you have created a DLL containing information about your business logic. This DLL can be used by your client application. In order to run the client application, the DLL must be included in the same folder in which the client application has been installed. This makes the assembly private to your application. Now suppose that the DLL needs to be reused in different applications. Therefore, instead of copying the DLL in every client application folder, it can be placed in the global assembly cache using the GAC tool. These assemblies are called shared assemblies.

4. Can one DLL file contain the compiled code of more than one .NET language?

No, a DLL file can contain the compiled code of only one programming language.

5. What is the maximum number of classes that can be contained in a DLL file?

There is no limit to the maximum number of classes that can be contained in a DLL file.

6. What is a satellite assembly?

Satellite assemblies are assemblies that are used to deploy language and culture specific resources for an application. In an application, a separate product ID is assigned to each language and a satellite assembly is installed in a language specific sub-directory.

7. Is versioning applicable to private assemblies?

No, versioning is not applicable to private assemblies as these assemblies reside in their individual folders. Versioning can be applied to GAC only.

8. What is metadata?

An assembly metadata describes every data type and member defined in the code. It stores the description of an assembly, such as name, version, culture, public key of an assembly along with the types exported, other assemblies dependent on this assembly, and security permissions needed to run the application. In addition, it stores the description of types, such as the name, visibility, base class, interfaces implemented, and members, such as methods, fields, properties, events, and nested types.

It also stores attributes. Metadata is stored in binary format. Therefore, metadata of an assembly is sharable among applications that execute on various platforms. It can also be exported to other applications to give information about the services and various features of an application.

9. What is Assembly Manifest?

Assemblies maintain all their information in a special unit called the manifest. Every assembly has a manifest.

The followings are the contents of an Assembly Manifest:

- **Assembly name** - Represents a text string that specifies the assembly's name.
- **Version number** - Represents a major and minor version number, as well as a revision and build number. The CL.R makes use of these numbers to enforce version policy.
- **Culture** - Represents information of the culture or language, which the assembly supports. An assembly is a container of only resources containing culture- or language-specific information.
- **Strong name information** - Represents the public key from the publisher, if a strong name is assigned to an assembly.
- **List of all files in the assembly** - Represents a hash of each file contained in the assembly and a file name.
- **Type reference information** - Represents the information used at the runtime to map a type reference to the file that contains its declaration and implementation.
- **Information on referenced assemblies** - Represents a list of other assemblies that are statically referenced by the assembly. Each reference includes the names of dependent assemblies, assembly metadata (version, culture, operating system, and so on), and public key, if the assembly is strong named.

10. What is the value of the Copy Local property when you add an assembly in the GAC?

False.

11. What is Native Image Generator?

The Native Image Generator (*Ngen.exe*) is a tool that creates a native image from an assembly and stores that image to native image cache on the computer. Whenever, an assembly is run, this native image is automatically used to compile the original assembly. In this way, this tool improves the performance of the managed application by loading and executing an assembly faster.

Note that native images are files that consist of compiled processor-specific machine code. The Ngen.exe tool installs these files on to the local computer.

12. Name the MSIL Disassembler utility that parses any .NET Framework assembly and shows the information in human readable format

The *Ildasm.exe* utility.

13. What is the significance of the Strong Name tool?

The Strong Name utility (*sn.exe*) helps in creating unique public-private key pair files that are called strong name files and signing assemblies with them. It also allows key management, signature generation, and signature verification.

14. How can different versions of private assemblies be used in the same application without a re-build?

You can use different versions of private assemblies in the same application without a re-build by specifying the assembly version in the *AssemblyInfo.cs* or *AssemblyInfo.vb* file.

15. What is Global Assembly Cache (GAC) ?

GAC is a central repository (cache) in a system in which assemblies are registered to share among various applications that execute on local or remote machines. .NET Framework provides the GAC tool (*gacutil.exe* utility), which is used to view and change the content of GAC of a system. Adding new assemblies to GAC and removing assemblies from GAC are some of the tasks that can be performed by using the *gacutil.exe* utility. GAC can contain multiple versions of the same .NET assembly. CLR checks GAC for a requested assembly before using information of configuration files.

The *gacutil.exe /i <assembly name>* - is the command that is used to install an assembly in GAC. Users use the Command Prompt of Visual Studio to install an assembly in GAC by using this command.

You can see all the assemblies installed in the GAC using the GAC viewer, which is located at the *<WinDrive>:<WinDir>\assembly directory*, where *<WinDir>* is windows in Windows XP or windows in Windows Vista or WinNT in Windows 2000. Apart from the list of assemblies, the assembly viewer also shows relevant information, such as the global assembly name, version, culture, and the public key token.

16. Where is the information regarding the version of the assembly stored?

Information for the version of assembly is stored inside the assembly manifest.

17. Discuss the concept of strong names.

Whenever, an assembly is deployed in GAC to make it shared, a strong name needs to be assigned to it for its unique identification. A strong name contains an assembly's complete identity - the assembly name, version number, and culture information of an assembly. A public key and a digital signature, generated over the assembly, are also contained in a strong name. A strong name makes an assembly identical in GAC.

18. What is the difference between .EXE and .DLL files?

EXE

1. It is an **executable file**, which can be run independently.
2. EXE is an out-process component, which means that it runs in a separate process.
3. It cannot be reused in an application.
4. It has a main function.

DLL

1. It is **Dynamic Link Library** that is used as a part of EXE or other DLLs. It cannot be run independently.
2. It runs in the application process memory, so it is called as in-process component.
3. It can be reused in an application.
4. It does not have a main function.

19. Which utility allows you to reference an assembly in an application?

An assembly can be referenced by using the *gacutil.exe* utility with the */r* option. The */r* option requires a reference type, a reference ID, and a description.

20. The AssemblyInfo.cs file stores the assembly configuration information and other information, such as the assembly name, version, company name, and trademark information. (True/False).

True.

Cloud Computing

1. What is cloud computing?

The **cloud computing** is the computing which is completely based on the Internet. It can also be defined as the next stage in the evolution of the Internet. The cloud computing uses the cloud (Internet) that provides the way to deliver the services whenever and wherever the user of the cloud needs. Companies use the cloud computing to fulfill the needs of their customers, partners, and providers. The cloud computing includes vendors, partners, and business leaders as the three major contributors. The vendors are the one who provide applications and their related technology, infrastructure, hardware, and integration.

The partners are those who offer cloud services demand and provide support service to the customers. The business leaders are the ones who use or evaluate the cloud service provided by the partners. The cloud computing enables the companies to treat their resources as a pool and not as independent resources.

2. What is a cloud?

A **cloud** is a combination of hardware, networks, storage, services, and interfaces that helps in delivering computing as a service. It has broadly three users which are end user, business management user, and cloud service provider. The end user is the one who uses the services provided by the cloud. The business management user in the cloud takes the responsibility of the data and the services provided by the cloud. The cloud service provider is the one who takes care or is responsible for the maintenance of the IT assets of the cloud. The cloud acts as a common center for its users to fulfill their computing needs.

3. What are the basic characteristics of cloud computing?

The four basic characteristics of cloud computing are given as follows:

- Elasticity and scalability.
- Self-service provisioning and automatic de-provisioning.
- Standardized interfaces.
- Billing self-service based usage model.

4. What is a Cloud Service?

A cloud service is a service that is used to build cloud applications. This service provides the facility of using the cloud application without installing it on the computer. It reduces the maintenance and support of the application as compared to those applications that are not developed using the cloud service. The different kinds of users can use the application from the cloud service, which may be public or private application.

5. What are main features of cloud services?

Some important features of the cloud service are given as follows:

- Accessing and managing the commercial software.
- Centralizing the activities of management of software in the Web environment.
- Developing applications that are capable of managing several clients.
- Centralizing the updating feature of software that eliminates the need of downloading the upgrades.

6. How many types of deployment models are used in cloud?

There are 4 types of deployment models used in cloud:

1. Public cloud
2. Private cloud
3. Community cloud
4. Hybrid cloud

7. What is the AppFabric component?

The **AppFabric** component is used to create access control and distribute messages across clouds and enterprises. It has a service-oriented architecture, and can be considered as the backbone of the **Windows Azure** platform. It provides connectivity and messaging among distributed applications. It also has the capabilities of integrating the applications and the business processes between cloud services and also between cloud services and global applications.

The **AppFabric** component provides a development environment that is integrated with Visual Studio 2010.The **Windows Communication Foundation (WCF)** services built in VS 2010 can be published on cloud from the Visual Studio design environment.

The two important services of AppFabric are as follows:

- **Access Control Service (ACS)** - Allows rules-driven and claims-based access control for distributed applications. These claims-based rules and authorization roles can be defined in the cloud for accessing on-premise and cloud services. The claim can be a user or application attribute, which the service application expects, such as e-mail address, phone number, password, and role, for appropriate access control. When any application wants to use the Web service, it sends the required claims to ACS for requesting a token. ACS converts the input claims into output claims by following the rules of mapping. These rules are created during the configuration of ACS. The ACS issues a token containing the output claims for the consumer application. This application uses this token in the request header and sends to the Web service. This service validates the claims in the token and gives suitable access to the user.
- **Service bus** - Provides messaging between cross-enterprise and cross-cloud scenarios. It provides publish/subscribe, point-to-point, and queues message patterns for exchange of messages across distributed applications in the cloud. It integrates with the Access Control service to establish secure relay and communication.

8. Why does an organization need to manage the workloads?

The workload can be defined as an independent service or a set of code that can be executed. It can be everything from a data-intensive workload to storage or a transaction processing workload and does not rely upon the outside elements. The workload can be considered as a small or complete application.

The organization manages workloads because of the following reasons:

- To know how their applications are running.
- To know what functions they are performing.
- To know the charges of the individual department according to the use of the service.

9. Which services are provided by Window Azure operating system?

Windows Azure provides three core services which are given as follows:

- Compute
- Storage
- Management

10. Explain hybrid and community cloud.

The **hybrid** cloud consists of multiple service providers. This model integrates various cloud services for Hybrid Web hosting. It is basically a combination of private and public cloud features. It is used by the company when a company has requirements for both the private and public clouds. Consider an example when an organization wants to implement the **SaaS (Software as a Service)** application throughout the company. The implementation requires security that can be provided by the private cloud used inside the firewall. The additional security can be provided by the VPN on requirement. Now, the organization has both the private and public cloud features.

The **community** cloud provides a number of benefits, such as privacy and security. This model, which is quite expensive, is used when the organizations having common goals and requirements are ready to share the benefits of the cloud service.

11. Explain public and private cloud.

The **public** cloud (or external cloud) is freely available for access. You can use a public cloud to collect data of the purchasing of items from a Web site on the Internet. You can also use public cloud for the reasons, which are given as follows:

- Helps when an application is to be used by a large number of people, such as an e-mail application, on the Internet.
- Helps when you want to test the application and also needs to develop the application code.

- Helps when you want to implement the security for the application.
- Helps when you want to increase the computing capacity.
- Helps when you are working on the projects in collaboration.
- Helps when you are developing the project on an ad-hoc basis by using PaaS.

The **private** cloud allows the usage of services by a single client on a private network. The benefits of this model are data security, corporate governance, and reliability concerns. The private cloud is used by the organization when it has a huge, well-run data center having a lot of spare capacity. It is also used when an organization is providing IT services to its clients and the data of organization is highly important. It is best suited when the requirements are critical.

The characteristics of this model are given as follows:

- Provides capability to internal users and allows provision of services.
- Automates the tasks of management and provides the billing of consumption of a particular service.
- Offers a well-managed environment.
- Enables the optimization of computational resources, such as servers.
- Manages the workload of the hardware.
- Offers self-service based provisioning of hardware resources and software.

12. Give a brief introduction of Windows Azure operating system.

The **Windows Azure** operating system is used for running cloud services on the Windows Azure platform, as it includes necessary features for hosting your services in the cloud. It also provides runtime environment that consists of Web server, computational services, basic storage, queues, management services, and load balancers. The operating system provides development. Fabric for development and testing of services before their deployment on the Windows Azure in the cloud.

13. What are the advantages of cloud services?

Some of the advantages of cloud service are given as follows: Helps in the utilization of

investment in the corporate sector; and therefore, is cost saving.

- Helps in the developing scalable and robust applications. Previously, the scaling took months, but now, scaling takes less time.
- Helps in saving time in terms of deployment and maintenance.