**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 object-invariants. 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 to 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. TheMemoryMappedFile.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 IDisposableinterface.

**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 aComplex object. The assigned value represents the real part of the complex number, and its imaginary part becomes0. 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 forSystem.Int32.