**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
5. **19. Which utility allows you to reference an assembly in an application?**
6. 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.
7. **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).**
8. True.