1. **What is the primary idea that underlies Dependency Injection?**

Dependency Injection is a technique that helps create flexible applications and simplifies unit testing by achieving Inversion of Control (IoC) between classes and their dependencies. It makes it easy to create loosely coupled components. Inversion of control (IoC) is a design principle in which custom-written portions of a computer program receive the flow of control from a generic, reusable library.

1. **Describe the role of interfaces in implementing a dependency injection container.**

By implementing a dependency injection container, components can consume functionality defined by interfaces without having any first-hand knowledge of which implementation classes are being used. Changing the components that implement the interfaces that define application features makes it easier to change the behavior of an application. This results in components that are easier to isolate for unit testing.

1. **List the three stages of building basic Ninject functionality, and briefly define each stage.**

First stage: Prepare Ninject for use - create an instance of a Ninject kernel.

Second stage: Configure the kernel - make Ninject understand which implementation objects you want to use for each interface.

Third Stage: Create an object - you must initialize to use it, you can't control lifetime easily, you can't make unit tests on it.

1. **What is the purpose of a custom dependency resolver?**

A custom dependency resolver ensures that the MVC Framework uses Ninject whenever it creates an object. This can be done by implementing the IDependencyResolver interface and registering an instance of the interface on application startup.

1. **What is the purpose of conditional binding? Describe how it works.**

Conditional binding specifies what class Ninject should use as implementation of an interface when creating a particular object. Data-binding expressions are contained within <%# and %> delimiters and use the Eval and Bind functions. You can call any publicly scoped code within the <%# and %> delimiters to execute that code and return a value during page processing.

1. **Why is setting the object scope important when using dependency injection?**

Setting the object scope is important when using dependency injection because it ensures that decoupling is accomplished and lets you control the lifecycle of the objects you create.

1. **Briefly describe the Test Driven Development approach in developing an application.**

Test Driven Development (TDD) is a development approach where a test is written first, then the code is written to fulfill the test, followed by refactoring. TDD allows you to catch bugs early and decouple code, making it more modular and scalable.

1. **Consider the Assert class. How do you access the methods of that class?**

The methods of the Assert class are static and are used to perform different kinds of comparison between the expected and actual results. Each Assert method allows different types of comparison to be made and throws an exception if the result is not what was expected. The exception is used to indicate that a test has failed. You access these with:

using Microsoft.VisualStudio.TestTools.UnitTesting;

1. **Why is it useful to mock objects when developing software?**

It is useful to mock objects when developing software because they simulate the functionality of real objects from your project in a specific and controlled way. A mocking framework makes it easier to create fake components to isolate parts of the application for unit testing.

1. **Briefly describe the two main issues that arise when attempting to do Unit Tests without the aid of some sort of mocking library.**

The first main issue: the tests will fail if the discount logic in the implementation changes.

The second main issue: if the test fails, you won't know where the problem is.