1. Six Combinations of Access Modifier Keywords

1. public: The type or member can be accessed by any other code in the same assembly or another assembly that references it.
2. private: The type or member can only be accessed by code in the same class or struct.
3. protected: The type or member can only be accessed by code in the same class, or in a class that is derived from that class.
4. internal: The type or member can be accessed by any code in the same assembly, but not from another assembly.
5. protected internal: The type or member can be accessed by any code in the same assembly, or by any derived class in another assembly.
6. private protected: The type or member can be accessed by types derived from the containing class, but only within its containing assembly.

2. Difference between static, const, and readonly Keywords

- static: Belongs to the type itself rather than to a specific object. Can be used with fields, methods, properties, events, and constructors.
- const: Constant value that cannot be changed. It is implicitly static and must be initialized as it is declared.
- readonly: Can only be assigned a value during declaration or in a constructor in the same class. It can be used with instance fields.

3. What Does a Constructor Do?

A constructor initializes an object of a class. It is called when an instance of the class is created. It can set default values for fields and perform other initialization tasks.

4. Why is the partial Keyword Useful?

The partial keyword allows the definition of a class, struct, or interface to be split into multiple files. This is useful for organizing code, especially in large projects, and for separating auto-generated code from custom code.

5. What is a Tuple?

A tuple is a data structure that can hold a fixed number of items, each of which can be of a different type. In C#, tuples provide a concise way to group multiple values together.

6. What Does the C# record Keyword Do?

The record keyword defines a reference type that provides built-in functionality for value equality, immutability, and non-destructive mutation. Records are used to create immutable data models with less boilerplate code.

7. What Do Overloading and Overriding Mean?

- Overloading: Defining multiple methods with the same name but different parameters (signature) within the same class.
- Overriding: Providing a new implementation of a method in a derived class that is already defined in the base class. The base method must be marked with the virtual, abstract, or override keyword.

8. Difference Between a Field and a Property

- Field: A variable that is declared directly in a class or struct.
- Property: Provides a way to read, write, or compute the value of a private field. Properties have accessors (get and set) that can contain logic.

9. How to Make a Method Parameter Optional

Use default parameter values. Example:
void ExampleMethod(int x, int y = 5) { }

Here, y is optional and defaults to 5 if not provided.

10. What is an Interface and How is it Different from an Abstract Class?

- Interface: Defines a contract that implementing classes must follow. It can only contain declarations of methods, properties, events, and indexers without any implementation.
- Abstract Class: Can have both abstract members (without implementation) and concrete members (with implementation). Abstract classes can provide some functionality while forcing derived classes to implement the rest.

11. Accessibility Level of Members of an Interface

All members of an interface are implicitly public.

True/False Questions

12. True. Polymorphism allows derived classes to provide different implementations of the same method.
13. True. The override keyword is used to indicate that a method in a derived class is providing its own implementation of a method.
14. False. The new keyword is used to hide a method in the base class, not to override it.
15. False. Abstract methods cannot be used in a normal (non-abstract) class.
16. True. Normal (non-abstract) methods can be used in an abstract class.
17. True. Derived classes can override methods that were virtual in the base class.
18. True. Derived classes can override methods that were abstract in the base class.
19. False. In a derived class, you cannot override a method that was neither virtual nor abstract in the base class.

20. False. A class that implements an interface must provide an implementation for all the members of the interface.
21. True. A class that implements an interface can have other members that aren’t defined in the interface.
22. False. A class can have only one base class (single inheritance).
23. True. A class can implement more than one interface.