Test your knowledge

1. What are the six combinations of access modifier keywords and what do they do?

Six Access Modifiers in C# and Their Functionality

Access Modifier	Accessibility
`public`	Accessible everywhere (inside and outside the class, even in different assemblies).
`private`	Accessible only within the same class.
`protected`	Accessible within the same class and derived (child) classes.
`internal`	Accessible within the same assembly (but not outside).
`protected internal`	Accessible within the same assembly and also in derived (child) classes in other assemblies.
`private protected`	Accessible only within the same class and derived (child) classes within the same assembly.

2. What is the difference between the static, const, and readonly keywords when applied to a type member?

static means the member belongs to the class itself, not instances, and is shared across all objects. const is a compile-time constant that must be assigned at declaration and cannot change. readonly is a runtime constant that can be assigned in a constructor or at declaration but cannot be modified afterward.

3. What does a constructor do?
A constructor initializes an object when it is created, setting default values and executing setup logic.

4. Why is the partial keyword useful?
The partial keyword allows splitting a class, struct, or interface across multiple files, improving code organization.

5. What is a tuple?
A tuple is a lightweight data structure that groups multiple values into a single object without defining a custom class.

6. What does the C# record keyword do?
The record keyword defines an immutable reference type with built-in value equality, useful for data models.

7. What does overloading and overriding mean?
Overloading allows multiple methods with the same name but different parameters, while overriding allows a derived class to provide a new implementation for a virtual or abstract method.

8. What is the difference between a field and a property?
A field is a direct variable inside a class, while a property provides controlled access to a field using get and set.

9. How do you make a method parameter optional?
Use default parameter values (void Method(int x = 10)) or optional parameters (void Method(int x, int y = 5)).

10. What is an interface and how is it different from abstract class?
An interface defines a contract with method signatures but no implementation, while an abstract class can have both abstract and implemented methods.

11. What accessibility level are members of an interface?
Interface members are always public by default and cannot have access modifiers.

12. True/False. Polymorphism allows derived classes to provide different implementations of the same method.
True – Polymorphism allows derived classes to provide different implementations of the same method.

13. True/False. The override keyword is used to indicate that a method in a derived class is providing its own implementation of a method.
True – The override keyword is used when a derived class provides its own implementation of a base class method.

14. True/False. The new keyword is used to indicate that a method in a derived class is providing its own implementation of a method.
True – The new keyword hides a method from the base class instead of overriding it.

15. True/False. Abstract methods can be used in a normal (non-abstract) class.
False – Abstract methods must be inside an abstract class.

16.True/False. Normal (non-abstract) methods can be used in an abstract class.
True – Normal (non-abstract) methods can be used in an abstract class.

17. True/False. Derived classes can override methods that were virtual in the base class.
True – Virtual methods in a base class can be overridden in a derived class.

18. True/False. Derived classes can override methods that were abstract in the base class.
True – Abstract methods in a base class must be overridden in a derived class.

19. True/False. In a derived class, you can override a method that was neither virtual non abstract in the base class.
False – You can only override methods marked as virtual or abstract in the base class.

20. True/False. A class that implements an interface does not have to provide an implementation for all of the members of the interface.
False – A class must implement all members of an interface it implements.

21. True/False. A class that implements an interface is allowed to have other members that aren’t defined in the interface.
True – A class implementing an interface can have additional members beyond those defined in the interface.

22. True/False. A class can have more than one base class.
False – C# does not support multiple inheritance, but a class can inherit from only one base class.

23. True/False. A class can implement more than one interface.
True – A class can implement multiple interfaces, allowing flexibility and multiple behaviors.