Test your knowledge

1. What are the six combinations of access modifier keywords and what do they do?
   Public: members can be accessed anywhere.
   Protected: members can be accessed in the current class and subclasses
   Internal: members can be accessed in the current assemly
   Private: members can only be accessed in the current class
   Protected Internal: members can be accessed in the same assembly or in a derived class in other assemblies.
   Private Protected: members can be accessed in the containing class or in a class that derives from a containing class, but only in the same assembly.
2. What is the difference between the static, const, and readonly keywords when applied to a type member?
   Static: is used to specify a static member, which means static members are common to all the objects and they do not tie to a specific object.
   Const: means constant. Constant fields or local variables must be assigned a value at the time of declaration, then after that, they cannot be modified. By default constant are static, hence you cannot define a constant type as static.
   Readonly: A readonly field can be initialized either at the time of declaration or within the constructor of the same class. Therefore, readonly fields can be used for run-time constants.
3. What does a constructor do?
   Constructor is used to create instance of the class, to initialize the fields.
4. Why is the partial keyword useful?
   - multiple developers can work simultaneously in the same class in different files.
   - we can split the UI of the design code and the business logic code to read and understand the code.
   - we can maintain the application in an efficient manner by compressing large classes into small ones.
5. What is a tuple?
   A tuple is a data structure that contains a sequence of elements of different data types.
6. What does the C# record keyword do?
   We can use the record keyword to define a reference type that provides built-in functionality for encapsulating data. To some extent, it is a combination of class and struct.
7. What does overloading and overriding mean?
   overriding: Methods in base class and its subclasses share the same method name and same input parameters
   overloading: Methods in same class share the same method name, but different input parameters
8. What is the difference between a field and a property?
   The difference between field and property in C# is that a field is a variable of any type that is declared directly in the class while property is a member that provides a flexible mechanism to read, write or compute the value of a private field.
9. How do you make a method parameter optional?
   Three ways:
   1. Using method overloading

   2. Giving the default value
   3. Using OptionalAttribute

10. What is an interface and how is it different from abstract class?

Abstract Class	Interface
It contains both declaration and definition part.	It contains only a declaration part.
Multiple inheritance is not achieved by abstract class.	Multiple inheritance is achieved by interface.
It contain constructor.	It does not contain constructor.
It can contain static members.	It does not contain static members.
It can contain different types of access modifiers like public, private, protected etc.	It only contains public access modifier because everything in the interface is public.
The performance of an abstract class is fast.	The performance of interface is slow because it requires time to search actual method in the corresponding class.
It is used to implement the core identity of class.	It is used to implement peripheral abilities of class.
A class can only use one abstract class.	A class can use multiple interface.
If many implementations are of the same kind and use common behavior, then it is superior to use abstract class.	If many implementations only share methods, then it is superior to use Interface.
Abstract class can contain methods, fields, constants, etc.	Interface can only contains methods, properties, indexers, events.
It can be fully, partially or not implemented.	It should be fully implemented.

11. What accessibility level are members of an interface?
    Public.
12. True/False. Polymorphism allows derived classes to provide different implementations of the same method.
    True.
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.
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.
    False.
15. True/False. Abstract methods can be used in a normal (non-abstract) class.
    False.
16. True/False. Normal (non-abstract) methods can be used in an abstract class.
    True.
17. True/False. Derived classes can override methods that were virtual in the base class.
    True.
18. True/False. Derived classes can override methods that were abstract in the base class.

True.

19. True/False. In a derived class, you can override a method that was neither virtual non abstract in the base class.
False.
20. True/False. A class that implements an interface does not have to provide an implementation for all of the members of the interface.
True.
21. True/False. A class that implements an interface is allowed to have other members that aren’t defined in the interface.
True.
22. True/False. A class can have more than one base class.
False.
23. True/False. A class can implement more than one interface.
True.