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

 **public**: The member is accessible from any other code in the same assembly or another assembly that references it.

 **private**: The member is accessible only within the body of the class or the struct in which it is declared.

 **protected**: The member is accessible within its class and by derived class instances.

 **internal**: The member is accessible only within files in the same assembly.

 **protected internal**: The member is accessible within its class, derived classes, and any class within the same assembly.

 **private protected**: The member is accessible within its class and derived classes within the same assembly.

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

a type member?

 **static**:

* Applied to types and members.
* Belongs to the type itself rather than to a specific object.
* Accessible using the type name.

 **const**:

* Compile-time constant.
* Value is set at compile time and cannot be changed.
* Implicitly static.

 **readonly**:

* Run-time constant.
* Value can be assigned at declaration or in a constructor.
* Cannot be changed after the initial assignment.

3. What does a constructor do?

A constructor initializes a new instance of a class. It sets default values and initializes any fields or properties when an object is created.

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. It is useful for:

* Organizing code into separate files for better manageability.
* Working with code generators where parts of the class can be auto-generated and the rest can be manually written.

5. What is a tuple?

A tuple is a data structure that can hold multiple values of different types. It is often used to return multiple values from a method.

6. What does the C# record keyword do?

The record keyword defines a reference type that provides built-in functionality for encapsulating data. Records provide value equality, immutability, and succinct syntax for data-oriented classes.

7. What does overloading and overriding mean?

 **Overloading**: Defining multiple methods with the same name but different signatures (parameters).

 **Overriding**: Providing a new implementation of a method in a derived class that is already defined in a base class, marked with the virtual keyword.

8. What is the difference between a field and a property?

 **Field**: A variable that is declared directly in a class or struct. It stores data directly.

 **Property**: A member that provides a flexible mechanism to read, write, or compute the value of a private field. Properties have get and set accessors.

9. How do you make a method parameter optional?

Use default parameter values: void SomeMethod(double input\_value = 3.14) { }

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

 **Interface**: A contract that defines a set of methods and properties but does not provide implementation.

* Cannot contain fields or implementation.
* A class can implement multiple interfaces.

 **Abstract Class**: A class that cannot be instantiated and may contain abstract methods with no implementation as well as implemented methods.

* Can contain fields, properties, and methods.
* A class can inherit only one abstract class.

11. What accessibility level are members of an interface?

Interface members are implicitly public and cannot have any other access modifier.

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.

false

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