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

public: public member can be accessible from any class, including classes from other assemblies.
protected internal: protected internal member can be accessible from classes in same assembly. And only derived class can access it if it's from different assemblies.
protected: protected member can be accessible in same class and derived class.(no matter same assembly or not.)
internal: internal member can only accessible in same assembly.
private protected: private protected member can only access in same class and derived class in same assembly.
private: private member can only accessible in same class.

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

static is member that belongs to the class itself instead of any instance.
const is member that value is unchangeable it's implicitly static.
readonly is used to define member whose value is fixed at runtime which means it cannot be changed after initialization.

3. What does a constructor do?

Is used to create an object of the class and initialize class members. If no constructor provided there's default one. It can be overloaded by cannot be inherited so it also cannot be overridden.

4. Why is the partial keyword useful?

Firstly, it allows a single type or method can be divided into multiple files also can divide large codebases split into smaller one which makes the code easier to understand and maintain. Secondly, with constraint mentioned above, it'll create more convenience when doing collaboration.

5. What is a tuple?

a syntax to group multiple data elements in a lightweight data structure.

6. What does the C# record keyword do?

it provides concise way to define immutable classes.

7. What does overloading and overriding mean?

overloading means multiple methods in the same class and share the same name , same access modifier but different input/output.
overriding means within base class and derived class, they share same method signature however derived class can have different implementations.

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

field is the variable declared in the class, accessible by their name and cannot be used to implement interfaces. property is a method that can access to private field(s), accessible with accessor methods and can be used to implement interface..

9. How do you make a method parameter optional?
specify the value of the parameter.

10. What is an interface and how is it different from abstract class?
it defines the contract that defines a set of methods, properties, events and indexers and then implemented by derive classes.
Inheritance supports multiple inheritance, cannot have instance constructor, by default all members abstract and public(abstract has abstract and concrete members) and it cannot have fields.

11. What accessibility level are members of an interface?
default is public

12. True/False. 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.

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.

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

16. True/False. 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.

18. True/False. Derived classes can override methods that were abstract in the base class.

19. True/False. In a derived class, you can override a method that was neither virtual non 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.

21. True/False. A class that implements an interface is allowed to have other members that aren't defined in the interface.

22. True/False. A class can have more than one base class.

23. True/False. A class can implement more than one interface.