1. What is the difference between value type and reference type variables? What is boxing and unboxing?

   Value type
   Stored in stack
   Contains the actual data
   Copying creates a new, independent copy of the value
   Default value is 0. False depending on data type
   Value type is removed automatically once not being used

   Reference type
   Stored in heap
   Contains a references meaning address to the data
   Copying creates a reference to the same data
   Default value is null unless specified
   Reference type is removed by garbage collection

   Boxing
   Converting a value type to reference type (eg int to object)
   Example ;
   Int num = 10;
   Object boxed= num; //stored in a reference type variable

   Unboxing
   Converting a reference type to a value type
   Object boxed = 10;
   Int num = (int)boxed

2. What is meant by the terms managed resource and unmanaged resource in .NET 4.
   Managed resource:
   - Resources that are managed by the .Net runtime CLR
   - Automatically cleaned up by Garbage Collector
   - Examples: memory, objects, string, arrays

   Unmanaged Resource:

- Resources are not managed by the .Net runtime CLR
- Requires explicit cleanup by developers
- File handles, database connection, network socks are examples

Managed resources are resources that are handled by the .NET runtime's garbage collector (GC). These are typically .NET objects that reside in the managed heap, such as strings, arrays, and instances of classes. Unmanaged resources are resources that are not controlled by the .NET garbage collector. These include system resources such as file handles, database connections, sockets, and memory allocated via interop with native code.

3. Whats the purpose of Garbage Collector in .NET?

   The Garbage Collector (GC) in .NET is an automatic memory management system that manages the allocation and deallocation of memory for managed objects. Its primary purpose is to free developers from the need to manually manage memory, reducing common programming errors such as memory leaks and invalid memory access

   Allocates and releases memory for managed objects automatically.
   Ensures that unused objects are removed from memory when they are no longer accessible. This way developers dont need to write code to free memory, reducing eros like forgetting to release memory.

4. What happens when you divide an int variable by 0?
   System.DivideByZeroException”

5. What happens when you divide a double variable by 0?
   Positive: Infinity
   Negative: -Infinity
   Zero: NaN

6. What happens when you overflow an int variable, that is, set it to a value beyond its range?
   In an unchecked context, if an integer overflows, it wraps around to the opposite end of its range. In a checked context, an OverflowException is thrown if an integer operation overflows.
   ```
   int max = 2147483647;
   int result = max + 1; // Overflow
   Console.WriteLine(result); // Output: -2147483648 (wraps around)
   ```
   If the number goes only until 0-9, then 9+1 = 0, like it goes in loop. So, it goes in min value loop.

7. What is the difference between x = y++; and x = ++y;?

x = y++;: The value of y is assigned to x, and then y is incremented (post-increment).
x = ++y;: The value of y is incremented first, and the updated value is assigned to x (pre-increment).

8. What is the difference between break, continue, and return when used inside a loop statement?
   break: Terminates the loop immediately.
   continue: Skips the rest of the current iteration and moves to the next iteration of the loop.
   return: Exits the method entirely, ending the loop and returning control to the caller.

9. What are the three parts of a for statement and which of them are required? All optional.
   Initialization, condition, iteration
10. What is the difference between the = and == operators?
    =:assignment
    ==: comparison

11. Does the following statement compile? for (; true;) ; the loop becomes an infinite loop
12. What does the underscore _ represent in a switch expression?
    In C#, the underscore (_) in a switch expression is a discard pattern. It represents a catch-all case that matches any value not explicitly handled by other patterns. It serves as the default case in a switch expression.
13. What interface must an object implement to be enumerated over by using the foreach statement?
    IEnumerable (or IEnumerable<T> for generics).

14. When to use String vs. StringBuilder in C# ?
    String is immutable but StringBuilder is mutable

15. What is the base class for all arrays in C#? System.array

16. How do you sort an array in C#?Array.sort()

17. What property of an array object can be used to get the total number of elements in an array? length

18. Can you store multiple data types in System.Array? nope
19. What's the difference between the System.Array.CopyTo() and System.Array.Clone()?
    CopyTo(): Copies elements into an existing array.
    Clone(): Creates a shallow copy of the array.

20. What are the six combinations of access modifier keywords and what do they do?
    public: Accessible from anywhere.
    private: Accessible only within the same class.
    protected: Accessible within the class and derived classes.
    internal: Accessible within the same assembly.
    protected internal: Accessible within the same assembly and derived classes.
    private protected: Accessible within the same class and derived classes in the same assembly.

21. What is the difference between the static, const, and readonly keywords when applied to a type member?
    static: Belongs to the class, not an instance.
    const: Compile-time constant; value cannot change.
    readonly: Run-time constant; can only be set during initialization or in a constructor.

22. What does a constructor do?
    Initializes an object when it is created.

23. Why is the partial keyword useful?
    Because it allows a class, struct, or interface to be split across multiple files

24. What is a tuple?
    A lightweight data structure to hold multiple values of different types

25. What does the C# record keyword do?
    Defines a class optimized for immutability and value based equality

26. What does overloading and overriding mean?
    Overloading : Same method, different parameters
    Overriding: Same method but has different implementation so child uses this method to let the computer know to use this method, instead of parent method

27. What is the difference between a field and a property?
    Field; a variable directly in class
    Property: is to get access to field through getters and setters

28. How do you make a method parameter optional?

you can make a method parameter optional by providing a default value for it
Public static void Opt(string name= “paige”, int age){
}
Opt(8);

29. What is an interface and how is it different from abstract class?
    Interface: only method definition, no method implementation
    Abstract: may or may not have method implementation and abstract members

30. What accessibility level are members of an interface? Public by default

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

32. 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
33. 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
34. True/False. Abstract methods can be defined in a normal (non-abstract) class. False
35. True/False. Normal (non-abstract) methods can be implemented in an abstract class. True
36. True/False. Derived classes can override methods that were virtual in the base class. True
37. True/False. Derived classes can override methods that were abstract in the base class. True
38. True/False. In a derived class, you can override a method that was neither virtual nor abstract in the base class. False
39. True/False. A class that implements an interface does not have to provide an implementation for all of the members of the interface. False
40. True/False. A class that implements an interface is allowed to have other members that aren’t defined in the interface. True
41. True/False. A class can have more than one base class. False
42. True/False. A class can implement more than one interface. True