Practical questions on **interfaces** in C#

---

1. Basic Interface Implementation

Create an interface `IVehicle` with the following members:

- A property `Speed` of type `int`.
- A method `Start()` that prints "Vehicle started".
- A method `Stop()` that prints "Vehicle stopped".

Create a class `Car` that implements the `IVehicle` interface. Write a program to test the class.

---

2. Multiple Interfaces

Create two interfaces:

- `IDrawable` with a method `Draw()`.
- `IResizable` with a method `Resize(int percentage)`.

Create a class `Circle` that implements both interfaces. Demonstrate how to use explicit interface implementation for each method.

3.

Write an interface `ICalculator` with methods `Add`, `Subtract`, `Multiply`, and `Divide`.
Create a class `SimpleCalculator` that implements the interface. Write a program to perform basic arithmetic operations using this class.

---

4. Using Interface with Polymorphism

Create an interface `IShape` with a method `CalculateArea()`.
Create classes `Rectangle` and `Circle` that implement `IShape`. Write a program to calculate and display the area of different shapes using a list of `IShape`.

---

5. Extending Interfaces

Create an interface `IPrintable` with a method `Print()`.
Create another interface `ISavable` that extends `IPrintable` and adds a method `Save()`.
Create a class `Document` that implements `ISavable`. Write a program to demonstrate the extended functionality.

---

6. Interface with Properties

Create an interface `IEmployee` with:

- A property `Name` (string).
- A property `Salary` (double) with a getter and setter.
- A method `DisplayDetails()`.

Create a class `Manager` that implements `IEmployee`. Write a program to demonstrate the use of properties defined in the interface.

---

7. Interface for Comparisons

Create an interface `IComparable` with a method `CompareTo(object other)`.
Create a class `Student` with properties `Name` and `Marks`, implementing `IComparable` to compare students based on marks.
Write a program to sort a list of students using this interface.

---

8. Interface with Generic Types

Create a generic interface `IRepository<T>` with methods:

- `Add(T item)`.
- `Remove(T item)`.
- `GetAll()` (returns `IEnumerable<T>`).

Create a class `ProductRepository` that implements `IRepository<Product>`. Write a program to manage a list of products.

---

9. Real-World Application: Payment System

Create an interface `IPaymentProcessor` with a method `ProcessPayment(double amount).` Implement two classes:

- `CreditCardPayment`: Processes payments using a credit card.
- `PayPalPayment`: Processes payments through PayPal.

Write a program that accepts the `IPaymentProcessor` interface to process payments dynamically.