Q1: Why can't a struct inherit from another struct or class in C#?
Because structs are **value types**, designed to be lightweight and stored on the stack. Inheritance would add overhead. They can only implement interfaces, not inherit from classes/structs.

Q2: How do access modifiers impact the scope and visibility of a class member?
Access modifiers define **who can access a member**:

- private: only inside the same class/struct.
- internal: within the same project/assembly.
- public: accessible everywhere.
  They enforce **security and proper encapsulation**.

Q3: Why is encapsulation critical in software design?
Encapsulation **hides implementation details** and provides controlled access through methods/properties. It improves **security, maintainability, and flexibility** of code.

Q4: What are constructors in structs?
Constructors in structs are **special methods** used to initialize fields when creating an object. They can be parameterized but **structs can’t define a default (parameterless) constructor**—the compiler provides one automatically.

Q5: How does overriding methods like ToString() improve code readability?
Overriding ToString() gives a **meaningful string representation** of an object instead of the default type name. This makes **debugging, logging, and output** more readable and user-friendly.