Part01

- **Question: Why can't a struct inherit from another struct or class in C#?**
  The struct in C# is a Value Type, which means it is copied by value, not by reference.
  For this reason:
  It cannot inherit from another struct

- **Question: How do access modifiers impact the scope and visibility of a class member?**
  Access Modifiers control who can access the members of a class (Fields, Methods, Properties):

  Private:
  Members are visible only within the same class.
  Cannot be accessed from any other class, even if it is derived.

  Protected:
  Members are visible within the same class and its subclasses only.
  Useful for inheritance without exposing members to the outside world.

  Internal:
  Members are visible within the same assembly only.
  Cannot be accessed from other projects.

  protected internal:
  Members are visible within the same assembly or to derived classes anywhere.

  private protected:
  Members are visible to derived classes within the same assembly only.

- **Question: Why is encapsulation critical in software design?**
  Encapsulation protects data, organizes code, makes the program more flexible and maintainable, and reinforces the principles of clean and safe programming

- **Question: what is constructors in structs?**
  Parameterized Constructors
  A struct can have constructors with parameters to set initial values for its fields.

  Default Constructor
  A struct cannot define a default constructor by itself.
  C# automatically provides a default constructor that initializes all fields to their default values (0 or null).

- **Question: How does overriding methods like ToString() improve code readability?**
  Allows displaying a clear and understandable textual representation of the object instead of the default class name.

  Makes it easier for the developer to quickly understand the object's data without accessing each field.

  Helps simplify debugging, as useful information is displayed directly when the object is printe

- **Question: How does memory allocation differ for structs and classes in C#?**
  Classes (Reference Types):
  Stored in the Heap (dynamic memory).
  The variable holding the object contains a reference to the object, not the object itself.
  Copying the variable copies the reference only, not the object.

  Structs (Value Types):
  Typically stored in the Stack (temporary memory) or inside the containing object.
  The variable representing a struct holds a copy of the values, not a reference.
  Copying the variable copies the entire value, not just the reference**.**

Part02

- **What is copy constructor?**
  A copy constructor is a special constructor that creates a new object as a copy of an existing object. It is used to duplicate objects with the same value

- **What is Indexer, when used, as business mention cases u have to utilize it?**
  An Indexer is a special property in a class that allows objects to be accessed like an array using [ ]. It is used when we want to access elements inside a class easily without defining separate methods.
  Business cases: managing a list of products, employees, or shopping cart items.

- **Summarize keywords we have learnt last lecture**
  Keywords learned:
  Class: Blueprint for objects; can have fields, methods, properties.
  Struct: Value type; stored in stack; copied by value.
  Access Modifiers: Control visibility of class/struct members (public, private, protected, internal).
  Inheritance: Mechanism where a class derives from another class and reuses its code.
  Encapsulation: Wrapping data and methods inside a class; usually with private fields and public getters/setters.