- **Why is it recommended to explicitly assign values to enum members in some cases?**

  To avoid unexpected changes when enum values are reordered or extended.
- **What happens if you assign a value to an enum member that exceeds the underlying type's range?**

  It causes a compilation error.
- **What is the purpose of the virtual keyword when used with properties?**

  It allows the property to be overridden in derived classes.
- **Why can’t you override a sealed property or method?**

  Because sealed prevents further overriding to maintain behavior consistency.
- **What is the key difference between static and object members?**

  Static members belong to the class itself, while object members belong to instances.
- **Can you overload all operators in C#? Explain why or why not.**

  No, only a predefined set of operators can be overloaded (e.g., +, -, *), not ones like = or ?..
- **When should you consider changing the underlying type of an enum?**

  When you need to save memory or restrict values to a smaller range.
- **Why can't a static class have instance constructors?**

  Because you cannot create instances of a static class.
- **What are the advantages of using Enum.TryParse over direct parsing with int.Parse?**

  It avoids exceptions and safely handles invalid input.
- **What is the difference between overriding Equals and == for object comparison in C# struct and class?**

  Equals checks value equality (can be overridden), while == checks reference equality by default for classes but can be overloaded.
- **Why is overriding ToString beneficial when working with custom classes?**

  To provide meaningful, readable output instead of the default class name.
- **Can generics be constrained to specific types in C#? Provide an example.**

  Yes, using where T : constraint (e.g., where T : class, where T : struct).
- **What are the key differences between generic methods and generic classes?**

  Generic methods are type-parameterized at the method level, while generic classes are parameterized at the class level and apply to all methods inside.

- **Why might using a generic swap method be preferable to implementing custom methods for each type?**

  It reduces code duplication and works for all types.
- **How can overriding Equals for the Department class improve the accuracy of searches?**

  It ensures logical equality (e.g., same department name) instead of default reference equality.
- **Why is == not implemented by default for structs?**

  Because structs are value types, and implementing == requires defining how to compare fields explicitly.