# Questions

## Part 1:-

1. ### Why is it recommended to explicitly assign values to enum members in some cases?

   Explicitly assigning values ensures compatibility with external systems (e.g., APIs or databases), improves readability, and provides control over the exact numeric representation of each enum member.

2. ### What happens if you assign a value to an enum member that exceeds the underlying type's range?

   A compilation error occurs if the value exceeds the range of the underlying type (e.g., byte supports values 0–255). However, if explicitly cast, it may lead to unintended results or runtime issues.

3. ### What is the purpose of the virtual keyword when used with properties?

   It allows the property to be overridden in derived classes, enabling polymorphic behavior.

4. ### Why can’t you override a sealed property or method?

   A sealed property or method explicitly prevents further overrides in derived classes, ensuring that its behavior remains fixed.

5. ### What is the key difference between static and object members?

   - **Static members** belong to the class itself and are shared across all instances.
   - **Object members** belong to a specific instance and have instance-specific values or behavior.

6. ### Can you overload all operators in C#? Explain why or why not.

   No, you cannot overload all operators. For instance, operators like = and ?: are not overloadable because their functionality is tightly coupled with the language semantics.

7. When should you consider changing the underlying type of an enum?

When the default int type does not suit the required range or memory efficiency, such as using byte for small ranges or long for large numeric values.

8. Why can't a static class have instance constructors?

tatic classes cannot be instantiated, so instance constructors are not applicable. They are designed solely to provide static members and methods.

9. What are the advantages of using Enum.TryParse over direct parsing with int.Parse?

- Avoids exceptions for invalid values.
- Provides a safe way to parse strings into enums with a boolean success indicator.

10. What is the difference between overriding Equals and == for object comparison in C# struct and class?

- Overriding Equals provides a custom definition of equality for instances.
- Overloading == defines a new behavior for the equality operator but requires both sides to match the operator's type.

11. Why is overriding ToString beneficial when working with custom classes?

It provides meaningful and human-readable string representations for instance, aiding debugging and logging.

12. Can generics be constrained to specific types in C#? Provide an example.

Yes, generics can be constrained using constraints like where T : SomeType.

```
public class Repository<T> where T : Entity
{
   public void Add(T entity) { /* logic */ }
}
```

### 13. What are the key differences between generic methods and generic classes?

- **Generic methods** define type parameters scoped only to the method.
- **Generic classes** define type parameters at the class level and share them across all methods and members of the class.

### 14. Why might using a generic swap method be preferable to implementing custom methods for each type?

It promotes reusability and reduces redundancy by providing a single method that works with any type.

### 15. How can overriding Equals for the Department class improve the accuracy of searches?

it allows the comparison of Department objects based on meaningful criteria, such as ID or name, rather than default reference equality.

### 16. Why is == not implemented by default for structs?

Implementing == requires knowledge of how to compare struct fields, which could vary significantly. By default, structs rely on Equals, and developers can define their custom equality logic.

## Part 2: -

### 1. What we mean by Generalization concept using Generics?

Generalization refers to creating reusable code that works with any data type, allowing flexibility and reducing duplication. For example, a generic list can store objects of any type without type-specific implementations.

### 2. What we mean by hierarchy design in real business?

Hierarchy design organizes entities in a structured way, such as employees under managers or products categorized into subcategories, enabling efficient management and logical flow of responsibilities.