1. What's Enum data type, when is it used? And name three common built-in enums used frequently?

An Enum (short for *Enumeration*) is a distinct data type in C# that allows you to define a set of named constants. It enhances code readability and makes the code more self-explanatory by replacing magic numbers with meaningful names.

Enums are used when a variable (especially in conditions or switches) can only take one out of a small set of possible values—for example: days of the week, order status, user roles, etc.

Common built-in enums include:

- DayOfWeek – Represents the days from Sunday to Saturday.
- ConsoleColor – Used to define foreground/background colors in the console.
- FileAccess – Specifies the access level (Read, Write, ReadWrite) for file operations.

---

2. What are scenarios to use string vs StringBuilder?

The string type in C# is immutable, meaning every modification creates a new string in memory. While this is fine for occasional use, it becomes inefficient in operations that involve many changes.

Use string when:

- You're dealing with small or few string manipulations.
- You want readable and simple code.
- The text content is mostly static.

Use StringBuilder when:

- You need to perform many modifications (append, insert, remove, etc.) on a string.
- You're building long strings in loops (e.g., generating HTML, logs, or reports).
- Performance and memory usage are critical.

---

3. What is meant by a user-defined constructor and its role in initialization?
A **user-defined constructor** is a special method written by the developer to initialize objects of a class with specific values when they are created. Unlike the default constructor (which initializes values with defaults), a user-defined constructor allows you to pass parameters and set object properties explicitly.

Its role in initialization:

- Sets initial values for object fields.
- Ensures the object starts in a valid state.
- Can perform custom logic during object creation.

Example use case:
A Student class where you want to create a new student with name and ID at the moment of instantiation.

---

4. Compare between Array and Linked List

Feature	Array	Linked List
Memory allocation	Contiguous memory	Non-contiguous (nodes linked by refs)
Access time	Fast (O(1) by index)	Slow (O(n) to access by index)
Insertion/Deletion	Expensive (need shifting)	Efficient (especially at head/tail)
Size	Fixed after creation	Dynamic; grows/shrinks as needed
Memory usage	Compact	More memory (due to node pointers)
Use cases	When random access is required	When frequent insertions/deletions

---