**1)** **Question: What is the difference between int.Parse and Convert.ToInt32 when handling null inputs?**

int.Parse throws an ArgumentNullException if the input is null.

Convert.ToInt32 returns 0 when the input is null.

**2)** **Question: Why is TryParse recommended over Parse in user-facing applications?**

Safely handles invalid inputs without throwing exceptions, returning false instead.

Simplifies error handling, making the code more efficient.

**3) Question: Explain the real purpose of the GetHashCode() method.**

The GetHashCode() method generates a numerical value representing an object's state, primarily for use in hash-based collections like Dictionary or HashSet. It enables efficient lookups by determining the storage bucket for an object. While not guaranteed unique.

**4) Question: What is the significance of reference equality in .NET?**

Reference equality in .NET checks if two object references point to the same memory location. It is important for determining object identity and is faster than value equality since it compares memory addresses, not contents.

**5) Question: Why string is immutable in C# ?**

Strings are immutable in C# to ensure thread safety and performance optimization, as they can be shared across multiple parts of a program without risk of modification. Immutability also allows for efficient memory usage and caching of string values.

**6) Question: How does StringBuilder address the inefficiencies of string concatenation?**

StringBuilder addresses inefficiencies by allowing mutable string operations, so it doesn't create new string objects with each modification. It appends text to a dynamically sized internal buffer, reducing memory allocation and improving performance for frequent concatenations.

**7) Question: Why is StringBuilder faster for large-scale string modifications?**

StringBuilder is faster for large-scale string modifications because it uses a mutable buffer to store the string, allowing in-place changes without creating new string objects each time.

**8) Question: Which string formatting method is most used and why?**

The most commonly used string formatting method in C# is String.Format() because It offers a flexible and readable way to insert variable values into a string. It allows for placeholders ({0}, {1}, etc.) and supports formatting options for different data types, making it ideal for creating formatted strings dynamically.

**9) Explain how StringBuilder is designed to handle frequent modifications compared to strings.**

StringBuilder is designed to handle frequent string modifications efficiently by using a mutable buffer rather than creating new string objects every time a modification is made, as is the case with immutable strings.

**2 - What’s Enum data type, when is it used? And name three common built\_in enums used frequently?**

Enum is a special data type in C# that defines a set of named constants. It is used when you want to represent a fixed set of related values (such as days of the week, months, or status codes) in a type-safe manner.

Three Common Built-in Enums:

ConsoleColor – Used to define colors for the console text or background.

DayOfWeek – Represents the days of the week (Sunday, Monday, etc.).

FileAttributes – Represents attributes of a file (ReadOnly, Hidden, System, etc.).

**3- what are scenarios to use string Vs StringBuilder?**

Use string when dealing with simple, immutable text or when the string doesn't change often. It’s efficient for small-scale string operations. Use StringBuilder when you need to modify a string frequently, such as in loops or large-scale string manipulations, to avoid memory overhead and improve performance. StringBuilder is optimized for performance in scenarios with many string changes.