Part 01 Problems

Question 1

What is the difference between int.Parse and Convert.ToInt32 when handling null inputs?

- **int.Parse(null) throws an ArgumentNullException because int.Parse does not accept null values.**
- **Convert.ToInt32(null) returns 0 instead of throwing an exception because it checks for null values and handles them gracefully.**

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

- **TryParse does not throw an exception on conversion failure; it returns a boolean indicating success or failure.**
- **This makes the application more stable and user-friendly because it can handle invalid input without crashing.**

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

- **GetHashCode() provides a numeric representation (usually unique or near-unique) of an object used mainly for hashing operations.**
- **It is primarily used to optimize lookups and storage in data structures like dictionaries and hash sets.**

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

- **When multiple references point to the same object in memory, changes via one reference affect the others.**
- **Reference equality means two references refer to the exact same memory location.**
- **This affects how objects are managed in memory, especially for reference types.**

5. Question: Why is string immutable in C#?

- **Strings are immutable, meaning any modification creates a new copy instead of changing the original.**
- **This immutability makes strings thread-safe, improves performance with caching, and reduces security issues.**
- **Also, since strings are often used as dictionary keys, immutability ensures stable hash codes.**

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

- **StringBuilder maintains a resizable internal buffer to modify text without creating a new string object each time.**
- **This reduces memory consumption and significantly improves performance especially with large or frequent string modifications.**

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

- **Because it does not create a new string copy on each modification but directly changes the internal buffer.**
- **Reducing allocations and copying makes operations faster and more memory-efficient than working with immutable strings.**

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

- **String interpolation ($) is the most used because:**
  - **It's more concise and readable than concatenation or string.Format.**
  - **It allows easy embedding of variables in strings.**
  - **It makes code easier to maintain and understand.**

9. Question: Explain how StringBuilder is designed to handle frequent modifications compared to strings.

- **StringBuilder is a mutable object with an internal storage that can be modified.**
- **It supports operations like append, replace, insert, and remove without creating multiple string copies.**
- **Unlike immutable strings, which require new copies on each change**

Part 02

2- What is the Enum data type, when is it used? And name three common built-in enums used frequently?

- **Enum is short for "Enumeration," which means a defined set of named constant values.**
- **When is it used?**
  Enum is used when you want to define a collection of fixed values, like days of the week, order status (Accepted, Rejected, Pending), or specific colors. Its purpose is to improve code clarity and safety by avoiding magic numbers or strings.
- **Three common built-in enums in .NET:**
  - **ConsoleColor — represents console text and background colors (e.g., Red, Blue, Green).**
  - **DayOfWeek — represents the days of the week (Sunday, Monday, etc.).**
  - **FileAccess — represents file access permissions (Read, Write, ReadWrite).**

3- Scenarios to use string vs StringBuilder

- **When to use string?**

- **When the text is small and requires few modifications.**
- **When you need immutable text that does not change after creation.**
- **When performance is not critical, and the modifications are minimal.**

- **When to use StringBuilder?**
  - **When you need to build or modify large strings or perform many repeated modifications (e.g., in loops).**
  - **When you want better performance and memory usage because StringBuilder modifies strings in place rather than creating new copies on every change.**