**ASSIGNMENT#2**

**Bcsf20a019:**

**Muhammad Saad**

Differentiate between Generic vs non -Generic Collection in C#.

Generic Collections:

Generic collections in C# are designed to work with specific data types, which means they are type-safe.

They allow you to specify the type of data that will be stored in the collection at the time of declaration.

Examples of generic collections include List<T>, Dictionary<TKey, TValue>, and Queue<T>.

Non-Generic Collections:

Non-generic collections in C# are not type-safe. They can store objects of any type because they are based on the object type.

They were commonly used in earlier versions of C# but are less preferred in modern programming due to the lack of type safety.

Examples of non-generic collections include ArrayList, Hash table, and Queue (non-generic).

What collection is suitable for Form Design entry based on the user needs vs Form design entries that are fixed?

Generic Collections:

Suitable for form design entries based on user needs.

If the form allows users to enter various types of data, and you want to maintain type safety, you should use generic collections. For example, you can use a List<string> to store user-entered names or a Dictionary<int, string> to store key-value pairs where keys are integers and values are strings.

Non-Generic Collections:

Suitable for form design entries that are fixed and don't require type safety.

If the form has a fixed set of fields or entries that are not expected to change types, non-generic collections can be used. However, this approach is less common in modern C# programming due to the benefits of type safety offered by generic collections.

In summary, when designing forms, it's generally a good practice to use generic collections for flexibility and type safety, especially if the form allows variable data types. Non-generic collections can be considered for simple scenarios where the data structure is fixed and type safety is not a primary concern.