**Name: Imran khan**

**Roll No: BCSF20A018**

**Assignment 2**

**Subject: EAD**

**Question1:**

**Differentiate between Generic vs non -Generic Collection in C#?  
Non-Generic Collections:**

Non-generic collections are collections that do not specify the type of elements they contain. They can hold any type of object. They use where dynamically changing data types needed.

**Real life example:**

Non-generic collections are like mixed bags where you can store different types of items. Imagine you have a bag, and you can throw in anything - a pen, a toy, a book, or even food. These collections are like that. You can put any kind of object into them. The problem is, when you take something out of the bag, you might not know what it is.

**Advantages:**

1. Flexibility
2. Easy to Use
3. Dynamism
4. Compatibility

**Code:**

ArrayList list = new ArrayList();

list.Add(1);

list.Add("hello");

list.Add(true);

we can store any type data in this.

**Generic Collections**

Generic collections in C# are classes and interfaces that allow you to specify the type of elements they can contain when you create an instance of the collection. They use in generic data structures.

**Real life example:**

Imagine you have a special bag designed just for one type of thing, like a bag only for toys or another bag only for books. Generic collections are like these specialized bags. You know exactly what you put in, and you know exactly what you get out.

In C#, generic collections are strongly typed, meaning they store specific types of items. If you create a generic list of integers, you can only put integers in it.

**Advantages:**

1. Type Safety
2. Performance
3. Reusability

**Code:**

List<int> numbers = new List<int>();

numbers.Add(1);

numbers.Add(2);

we can only store integers in this.

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

**Dynamic Form Entries (Based on User Needs):**

For forms where the number and types of entries are determined by user needs and can change dynamically, a suitable collection is a dynamic list or collection. In C#, you can use a **List** or a similar dynamically resizable collection.

Here's why it's suitable:

1. **Dynamic Resizing:** Lists and similar collections can grow or shrink as needed. This is important when users can add or remove form fields dynamically.
2. **Flexibility:** You can add, modify, or remove elements easily based on user interactions. This allows you to adapt to changing user requirements.
3. **Ease of Iteration:** Lists provide straightforward ways to iterate through form entries, making it easy to process and validate user input.

**Form with Fixed Entries:**

For forms where the number and types of entries are fixed and do not change, we can use a collection that matches the fixed structure of our form. we might use **array**s or a collection of custom objects.

Here's why it's suitable:

1. **Predictability:** When form entries are fixed, using arrays or a predefined collection structure provides clarity and predictability.
2. **Type Safety:** we know exactly what types of data to expect, which leads to type safety in our code.
3. **Validation:** Fixed form structures make it easier to define validation rules and constraints.