Creating a lab exercise focused on practicing C# collections and generics can help learners understand how to work with data in a more flexible and type-safe manner. Here's a detailed lab exercise that includes objectives, prerequisites, tasks, and a bonus challenge. This lab will cover various collection types such as `List<T>`, `Dictionary<TKey, TValue>`, and others, providing a comprehensive learning experience.

This lab exercise offers a hands-on approach to learning about C# collections and generics, providing a solid foundation for building more complex data structures and algorithms in the future.

Lab Exercise: Mastering C# Collections and Generics

Objectives:

- Understand and implement generic collections in C#.

- Learn to use different collection types such as `List<T>`, `Dictionary<TKey, TValue>`, `Queue<T>`, and `Stack<T>`.

- Practice iterating over collections and manipulating data within them.

Prerequisites:

- Basic knowledge of C# and its syntax.

- Familiarity with loops and conditional statements in C#.

- Visual Studio or any preferred IDE that supports C# development.

Tasks:

Task 1: Working with List<T>

1. Create a new console application in C#.

2. Implement a method `CreateAndPopulateList` that creates a `List<string>` and populates it with at least 10 names.

3. Implement another method `DisplayListContents` to iterate over this list and display each name on the console.

4. Add functionality to `InsertName` method that inserts a new name at a specified index within the list.

Task 2: Exploring Dictionary<TKey, TValue>

1. Create a method `CreateAndPopulateDictionary` that creates a `Dictionary<int, string>` where the key is a unique identifier (ID) and the value is a person's name. Populate it with at least 5 entries.

2. Implement `FindNameById` method that takes an ID as input and prints the corresponding name from the dictionary. Handle cases where the ID does not exist in the dictionary gracefully.

3. Add functionality to update a name for a given ID in the dictionary.

Task 3: Utilizing Queue<T> and Stack<T>

1. Create a `Queue<string>` and a `Stack<string>`. Populate both with at least 5 values (e.g., numbers or names).

2. Implement methods to demonstrate enqueueing/dequeueing from the queue and pushing/popping from the stack.

3. Show how to iterate over both collections without removing their elements.

Bonus Challenge:

- Implement a custom generic collection by defining a class `MyCollection<T>` that implements the `IEnumerable<T>` interface. Provide methods for adding, removing, and iterating over items in your custom collection.

- Add sorting functionality to your custom collection without using the built-in sort methods.

Tips:

- Pay attention to the capacity and count properties of collections and how they affect performance.

- Explore the various methods provided by each collection type to manipulate and query data.