Task

1. **Generics**
   - Create a generic class Box<T> that can store any type of item. The class should have methods AddItem(T item) and GetItem().
   - Demonstrate the usage of this class with different data types (e.g., int, string).
2. **List vs ArrayList:**
   - Create a non-generic ArrayList and add elements of different types.
   - Demonstrate how to retrieve elements from the ArrayList and explain boxing and unboxing with examples.
   - Create a generic List and enforce type safety by only allowing elements of a specified type.
3. **HashTable vs Dictionary vs SortedList:**
   - Create a Hashtable, Dictionary, and SortedList, each containing key-value pairs.
   - Populate each collection with data and print the elements to observe any differences in how they handle data, especially regarding ordering.
4. **Stack vs Queue:**
   - Implement a Stack, push elements onto it, and pop them to illustrate LIFO (Last In, First Out) behavior.
   - Implement a Queue, enqueue elements, and dequeue them to demonstrate FIFO (First In, First Out) behavior.

5. **Tuple vs ValueTuple:**
    - Create and use an old-style Tuple to store and retrieve multiple data items.
    - Create and use a ValueTuple, utilizing named parameters and demonstrating the advantages of this newer data structure.
6. **IEnumerator and IEnumerable**

    Description: Create a custom class that represents a data collection (e.g., a list of students). Implement the IEnumerable and IEnumerator interfaces to enable iteration over the data.

    - **Requirements:**
        - Implement GetEnumerator method in your custom class.
        - Implement MoveNext(), Reset(), and Current in your enumerator.
        - Test the implementation with a foreach loop.
7. **Creating an Indexer**

    Description: Add an indexer to the custom class you created in the previous question. The indexer should allow you to access elements in the collection using an index.

    - **Requirements:**
        - Implement a get and set method for the indexer.
        - Demonstrate retrieving and modifying elements using the indexer.