CUSTOM LINKED LIST

Ultimate C# Masterclass Assignment

Overview

The purpose of this assignment is to create a custom linked list data structure.	Library

The list must be able to store items of any type. Null values can also be stored in this list. It may be either a singly-linked list or a doubly-linked list. The solution of this assignment presented in the course will focus on a singly-linked list.

Required operations

The collection should implement the following interface:

```
public interface ILinkedList<T> : ICollection<T>
{
  void AddToFront(T item);
  void AddToEnd(T item);
}
```

This interface extends the ICollection<T>, so methods from this interface will need to be implemented as well. See the next page for the full list of required methods and their descriptions.

void AddToFront(T? item);

Adds a new item to the front of the list.

void AddToEnd(T? item);

Adds a new item to the end of the list.

int Count { get; }

Gets the count of all items currently stored in the list.

bool IsReadOnly { get; }

Gets a boolean saying if this list is readonly. It is not, so it should simply return false.

void Add(T? item);

Adds a new item to the end of the list, the same as the AddToEnd method.

bool Contains(T? item);

Returns true if the given item is present in the list, and false otherwise.

void CopyTo(T?[] array, int arrayIndex);

Copies all items from the list into a given array, starting at the given index of the array. Should throw exceptions if the given arguments are invalid (the array is null or not long enough to fit the items, or the index is invalid).

bool Remove(T? item);

Removes the first occurrence of the given item. Returns true if removed and false if no such item is present in the list.

IEnumerator<T?> GetEnumerator();

Returns the generic enumerator that will enable iteration of the collection with the foreach loop.

IEnumerator GetEnumerator();

Returns the enumerator that will enable iteration of the collection with the foreach loop.