# Lab: Linear Data Structures

This document defines the **in-class exercises** assignments the ["Data Structures" course @ Software University](https://softuni.bg/opencourses/data-structures). You can submit your code in the SoftUni Judge System - <https://judge.softuni.bg/Contests/574/Linear-Data-Structures-Stacks-and-Queues-Lab>.

## LinkedList<T>

Implement a data structure LinekdList<T> that holds a sequence of elements of generic type T. It should hold a **sequence of items in a sequence of linked nodes**. The list should support the following operations:

* int Count 🡪 returns the number of elements in the structure
* void AddFirst(T item) 🡪 adds an element to the start of the sequence
* void AddLast(T item) 🡪 adds an element to the end of the sequence
* T RemoveFirst() 🡪 removes an element from the start of the sequence and returns the element
* T RemoveLast() 🡪 removes an element from the end of the sequence and returns the element
* IEnumerable<T> 🡪 implement interface

RemoveFirst() and RemoveLast() methods should throw InvalidOperationException if the list is empty

### Examples



### Solution

Start by defining the class LinkedList<T>, you can define the Node class inside (in which case it doesn't need to be generic, as you will use T from the LinkedList)

Also, note that we are going to keep a reference to both the head and the tail of the list



First, implement AddFirst(). The utility method IsEmpty() checks if the Count is 0



AddLast() is almost the same



Remove methods are a little bit more complicated. Start with RemoveFirst()



Method RemoveLast() uses a utility method GetSecondToLast() which gets the second to last element. Try to implement it yourself



The last thing to do is to implement IEnumerable<T>



You need to implement two methods. The first is the actual that will do the work. The second one calls the first and you need it only for compatibility reasons

