**.NET impliments a doubly linked list.**

**There are more methods to call than what was manually implemented in the project.**

**My methods below apper on the surface to re-impliment the same methods in the .NET class.**

public void AddFirst(T value)

public void AddFirst(LinkedListNode<T> node)

public void AddLast(T value)

public void RemoveFirst()

public void Remove()

public void RemoveLast()

public void Add(T item)

public bool Contains(T item)

public void CopyTo(T[] array, int arrayIndex)

public bool Remove(T item)

public bool IsReadOnly

public void Clear()

IEnumerable<T>.GetEnumerator()

**Microsoft provide more methods in the .NET class such as:**

AddAfter(LinkedListNode<T>, T) to add a new node after a specific existing node.

AddBefore(LinkedListNode<T>, T) to add a new node before a specific existing node.

FindLast(T) to find the last node that has the a specific value.

GetHasCode() to get the has value for the object.

GetObjectData(SerializationInfo, StreamingContext) to serialise the linked list.

GetType() determine the data type in the linked list.

MemeberwiseClone() non-recursively-copies the current object.

OnDeserialization(Object) provides an alert when deserialisation has competed.

**Doubly Linked List advantages:**

The ability to traverse the list in reverse.

If the list is large, travsersing from the end can be advantagous to save time and processing power when it is known that the required node is closer to the end than the start.

The ability for a node to see what node came before may also help to move up and down the list.