# Linked List

A linked list is a data structure where data is stored on nodes that connect to one another sequentially but not contiguous in memory. Each node points to the next element in the structure. A linked list can either be singly linked, wherein each node only points to the next node after it, or double linked, wherein each node points to both the previous and next node. An advantage to using a linked list is that inserting or deleting elements at the beginning or the end of the data set is very fast. A disadvantage is that random access is not allowed. In order to retrieve an element, the entire list has to be traversed until the element is found. The following code snippet shows how to declare and initialize a linked list in C#, along with some common methods used.

```
public class Example
{
    public static void Main()
    {
        // Create the link list.
        string[] words =
            { "the", "fox", "jumps", "over", "the", "dog" };
        LinkedList<string> sentence = new LinkedList<string>(words);
        Display(sentence, "The linked list values:");
        Console.WriteLine("sentence.Contains(\"jumps\") = {0}",
            sentence.Contains("jumps"));

        // Add the word 'today' to the beginning of the linked list.
        sentence.AddFirst("today");
        Display(sentence, "Test 1: Add 'today' to beginning of the list:");

        // Move the first node to be the last node.
        LinkedListNode<string> mark1 = sentence.First;
        sentence.RemoveFirst();
        sentence.AddLast(mark1);
        Display(sentence, "Test 2: Move first node to be last node:");

        // Change the last node to 'yesterday'.
        sentence.RemoveLast();
        sentence.AddLast("yesterday");
        Display(sentence, "Test 3: Change the last node to 'yesterday':");

        // Move the last node to be the first node.
        mark1 = sentence.Last;
        sentence.RemoveLast();
        sentence.AddFirst(mark1);
        Display(sentence, "Test 4: Move last node to be first node:");

        // Indicate the last occurence of 'the'.
        sentence.RemoveFirst();
        LinkedListNode<string> current = sentence.FindLast("the");
        IndicateNode(current, "Test 5: Indicate last occurence of 'the':");

        // Add 'lazy' and 'old' after 'the' (the LinkedListNode named current).
        sentence.AddAfter(current, "old");
        sentence.AddAfter(current, "lazy");
        IndicateNode(current, "Test 6: Add 'lazy' and 'old' after 'the':");

        // Indicate 'fox' node.
        current = sentence.Find("fox");
        IndicateNode(current, "Test 7: Indicate the 'fox' node:");

        // Add 'quick' and 'brown' before 'fox':
        sentence.AddBefore(current, "quick");
        sentence.AddBefore(current, "brown");
        IndicateNode(current, "Test 8: Add 'quick' and 'brown' before 'fox':");

        // Keep a reference to the current node, 'fox',
        // and to the previous node in the list. Indicate the 'dog' node.
        mark1 = current;
        LinkedListNode<string> mark2 = current.Previous;
        current = sentence.Find("dog");
        IndicateNode(current, "Test 9: Indicate the 'dog' node:");
    }
}
```