```
using DoublyLinkedListWithErrors;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System.Collections.Generic;

namespace TestProject1
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestMethod1_addToHead1()
        {
            //When dll is empty
            DLLNode p = new DLLNode(33);
            DLList l = new DLList();
            l.addToHead(p);

            Assert.AreEqual(l.head.num, 33);
            Assert.AreEqual(l.tail.num, 33);
        }
        [TestMethod]
        public void TestMethod2_addToHead2()
        {
            //When dll is with only 1 node
            DLLNode p = new DLLNode(33);
            DLLNode q = new DLLNode(34);
            DLList l1 = new DLList();

            l1.addToHead(q);
            l1.addToHead(p);

            Assert.AreEqual(l1.head.num, 33);
            Assert.AreEqual(l1.tail.num, 34);
        }
        [TestMethod]
        public void TestMethod3_addToHead3()
        {
            //When dll is not empty with more than 1 nodes
            DLList l1 = new DLList();
            DLLNode p = new DLLNode(33);
            DLLNode q = new DLLNode(34);
            DLLNode x = new DLLNode(35);


            l1.addToHead(p);
            l1.addToHead(q);
            l1.addToHead(x);
            Assert.AreEqual(l1.head.num, 35);
        }
        [TestMethod]
        public void TestMethod4_addToTail1()
        {
            //test if statement. When the dll is empty
            DLList l = new DLList();
            DLLNode p = new DLLNode(33);

            l.addToTail(p);

            Assert.AreEqual(l.head.num, 33);
            Assert.AreEqual(l.tail.num, 33);
        }
        [TestMethod]
        public void TestMethod5_addToTail2()
        {
            //We know that add to head works. test add tails's if else
              statement. When the dll is not empty with only 1 node
            DLList l1 = new DLList();
            DLLNode p = new DLLNode(33);
            DLLNode q = new DLLNode(34);


            l1.addToTail(p);
            l1.addToTail(q);

            Assert.AreEqual(l1.head.num, 33);
            Assert.AreEqual(l1.tail.num, 34);
        }
        [TestMethod]
        public void TestMethod6_addToTail3()
        {
            //We know that add to head works. test add tails's if else
              statement. When the dll is not empty with more than 1 nodes
            DLLNode p = new DLLNode(33);
            DLLNode q = new DLLNode(34);
            DLLNode x = new DLLNode(35);
            DLList l1 = new DLList();

            l1.addToTail(p);
            l1.addToTail(q);
            l1.addToTail(x);

            Assert.AreEqual(l1.head.num, 33);
            Assert.AreEqual(l1.tail.num, 35);
        }


        [TestMethod]
        public void TestMethod7_removHead1()
        {
            //when there are more than 1 nodes
            DLLNode p = new DLLNode(33);
            DLLNode q = new DLLNode(34);
            DLList l1 = new DLList();

            l1.addToHead(q);
            l1.addToHead(p);

            l1.removHead();
            Assert.AreEqual(l1.head.num, 34);
        }
        [TestMethod]
        public void TestMethod8_removHead2()
        {
            //when there is only 1 node
            DLLNode p = new DLLNode(33);
            DLList l1 = new DLList();

            l1.addToHead(p);
            Assert.AreEqual(l1.head.num, 33);
            l1.removHead();
            Assert.IsTrue(l1.head== null);
        }
        [TestMethod]
        public void TestMethod9_removHead3()
        {
            //When the dll is empty
            DLList l = new DLList();

            l.removHead();
            Assert.AreEqual(l.head, null);
        }
        //Remove Tail
        [TestMethod]
        public void TestMethod10_removeTail1()
        {
            //when there are more than 1 nodes
            DLLNode p = new DLLNode(33);
            DLLNode q = new DLLNode(34);
            DLList l1 = new DLList();
            l1.addToHead(q);
            l1.addToHead(p);

            l1.removeTail();

            Assert.AreEqual(l1.tail.num, 33);
        }
        [TestMethod]
        public void TestMethod11_removeTail2()
        {
            //when there is only 1 node
            DLLNode p = new DLLNode(33);
            DLList l1 = new DLList();

            l1.addToHead(p);
            //Assert.AreEqual(l1.tail.num, 33); //Pre-check
            l1.removeTail();

            Assert.AreEqual(l1.tail, null);
        }
        [TestMethod]
        public void TestMethod12_removTail3()
        {
            //When the dll is empty
            DLList l = new DLList();

            l.removeTail();

            Assert.IsTrue(l.tail== null);
        }

        //search

        [TestMethod]
        public void TestMethod13_search1()
        {
            //1 node DLL
            DLLNode p = new DLLNode(33);
            DLList l = new DLList();

            l.addToHead(p);
            DLLNode q = l.search(33);

            Assert.AreEqual(p, q);
        }



        [TestMethod]
        public void TestMethod14_search2()
        {
            //more than 1 nodes DLL with 1 node meet the requirement
            DLLNode p = new DLLNode(33);
            DLLNode q = new DLLNode(34);
            DLList l = new DLList();

            l.addToHead(q);
            l.addToHead(p);
            DLLNode x = l.search(33);

            Assert.AreEqual(x, p);
        }

        [TestMethod]
        public void TestMethod15_search3()
        {
            //more than 1 nodes DLL with 1 node meet the requirement
            DLLNode p = new DLLNode(33);
            DLLNode q = new DLLNode(34);
            DLList l = new DLList();

            l.addToHead(q);
            l.addToHead(p);
            DLLNode x = l.search(34);

            Assert.AreEqual(x, q);
        }
        [TestMethod]
        public void TestMethod16_search4()
        {
            //not found
            DLLNode p = new DLLNode(33);
            DLLNode q = new DLLNode(34);
            DLList l = new DLList();

            l.addToHead(q);
            l.addToHead(p);
            DLLNode x = l.search(35);

            Assert.AreNotEqual(x, q);
            Assert.AreNotEqual(x, p);
        }


        /* [TestMethod]
           public void TestMethod15_search3()
           {
               //more than 1 nodes DLL with more than 1 nodes meet the
                 requirement

               var pointers = new List<int>();
               var pointers_expected = new List<int>();

               pointers_expected.Add(1);
               pointers_expected.Add(0);

```

```
                DLLNode p = new DLLNode(33);
                DLLNode q = new DLLNode(33);
                DLList l = new DLList();
                l.addToHead(q);
                l.addToHead(p);

                while q

                int pointer = l.search(33);
                Assert.AreEqual(1, pointer);

            }
        */
        /*
        public void TestMethod15_search3()
        {
            //searched item not in the dll !!!!!!
            DLLNode p = new DLLNode(33);
            DLLNode q = new DLLNode(34);
            DLList l = new DLList();

            l.addToHead(q);
            l.addToHead(p);
            bool pointer = l.search(32);

            Assert.isTrue(pointer);

        }
        */
        //Remove Node

        [TestMethod]
        public void TestMethod16_removeNode1()
        {
            //Not inside !!!!!!
            DLLNode p = new DLLNode(33);
            DLLNode q = new DLLNode(34);
            DLList l = new DLList();

            l.addToHead(p);
            l.removeNode(q);

            Assert.AreEqual(l.head, p);
        }
        [TestMethod]
        public void TestMethod17_removeNode2()
        {
            //Node inside, 1 Node only
            DLLNode p = new DLLNode(33);
            DLList l = new DLList();

            l.addToHead(p);
            l.removeNode(p);

            Assert.AreEqual(l.head, null);
            //Assert.AreEqual(p, l);
        }
        [TestMethod]
        public void TestMethod18_removeNode3()
        {
            //Node inside, more than 1 Node, head
            DLLNode p = new DLLNode(33);
            DLLNode q = new DLLNode(34);
            DLList l = new DLList();

            l.addToHead(q);
            l.addToHead(p);

            l.removeNode(p);

            Assert.AreEqual(l.head, q);
        }
        [TestMethod]
        public void TestMethod19_removeNode4()
        {
            //Node inside, more than 1 Node, tail
            DLLNode p = new DLLNode(33);
            DLLNode q = new DLLNode(34);
            DLList l = new DLList();

            l.addToHead(q);
            l.addToHead(p);
            l.removeNode(q);

            Assert.AreEqual(l.head, p);
        }

        //Total
        [TestMethod]
        public void TestMethod20_Total1()
        {
            //0 Nodes for total
            DLList l = new DLList();

            Assert.AreEqual(l.total(), 0);
        }
        [TestMethod]
        public void TestMethod21_Total2()
        {
            // 1 Node for total
            DLLNode p = new DLLNode(33);
            DLList l = new DLList();
            l.addToTail(p);

            Assert.IsTrue(l.total()==33);
        }
        [TestMethod]
        public void TestMethod22_Total3()
        {
            // > 1 Nodes for total
            DLLNode p = new DLLNode(33);
            DLLNode q = new DLLNode(34);
            DLList l = new DLList();
            l.addToHead(p);
            l.addToHead(q);

            Assert.IsTrue(l.total()==67);
        }
    }
}
```