

```
1 using DoublyLinkedListWithErrors;
2 using Microsoft.VisualStudio.TestTools.UnitTesting;
3 using System.Collections.Generic;
4
5 namespace TestProject1
6 {
7     [TestClass]
8     public class UnitTest1
9     {
10         [TestMethod]
11         public void TestMethod1_addToHead1()
12         {
13             //When dll is empty
14             DLLNode p = new DLLNode(33);
15             DLLList l = new DLLList();
16             l.addToHead(p);
17
18             Assert.AreEqual(l.head.num, 33);
19             Assert.AreEqual(l.tail.num, 33);
20         }
21         [TestMethod]
22         public void TestMethod2_addToHead2()
23         {
24             //When dll is with only 1 node
25             DLLNode p = new DLLNode(33);
26             DLLNode q = new DLLNode(34);
27             DLLList l1 = new DLLList();
28
29             l1.addToHead(q);
30             l1.addToHead(p);
31
32             Assert.AreEqual(l1.head.num, 33);
33             Assert.AreEqual(l1.tail.num, 34);
34         }
35         [TestMethod]
36         public void TestMethod3_addToHead3()
37         {
38             //When dll is not empty with more than 1 nodes
39             DLLList l1 = new DLLList();
40             DLLNode p = new DLLNode(33);
41             DLLNode q = new DLLNode(34);
42             DLLNode x = new DLLNode(35);
43
44
45             l1.addToHead(p);
46             l1.addToHead(q);
47             l1.addToHead(x);
48             Assert.AreEqual(l1.head.num, 35);
49         }
50     }
51 }
```

```
50     [TestMethod]
51     public void TestMethod4_addToTail1()
52     {
53         //test if statement. When the dll is empty
54         DLLList l = new DLLList();
55         DLLNode p = new DLLNode(33);
56
57         l.addToTail(p);
58
59         Assert.AreEqual(l.head.num, 33);
60         Assert.AreEqual(l.tail.num, 33);
61     }
62     [TestMethod]
63     public void TestMethod5_addToTail2()
64     {
65         //We know that add to head works. test add tails's if else statement. When the dll is not empty with only 1 node
66         DLLList l1 = new DLLList();
67         DLLNode p = new DLLNode(33);
68         DLLNode q = new DLLNode(34);
69
70
71         l1.addToTail(p);
72         l1.addToTail(q);
73
74         Assert.AreEqual(l1.head.num, 33);
75         Assert.AreEqual(l1.tail.num, 34);
76     }
77     [TestMethod]
78     public void TestMethod6_addToTail3()
79     {
80         //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
81         DLLNode p = new DLLNode(33);
82         DLLNode q = new DLLNode(34);
83         DLLNode x = new DLLNode(35);
84         DLLList l1 = new DLLList();
85
86         l1.addToTail(p);
87         l1.addToTail(q);
88         l1.addToTail(x);
89
90         Assert.AreEqual(l1.head.num, 33);
91         Assert.AreEqual(l1.tail.num, 35);
92     }
93
94
95     [TestMethod]
96     public void TestMethod7_removHead1()
```

```
97     {
98         //when there are more than 1 nodes
99         DLLNode p = new DLLNode(33);
100         DLLNode q = new DLLNode(34);
101         DLLList l1 = new DLLList();
102
103         l1.addToHead(q);
104         l1.addToHead(p);
105
106         l1.removHead();
107         Assert.AreEqual(l1.head.num, 34);
108     }
109     [TestMethod]
110     public void TestMethod8_removHead2()
111     {
112         //when there is only 1 node
113         DLLNode p = new DLLNode(33);
114         DLLList l1 = new DLLList();
115
116         l1.addToHead(p);
117         Assert.AreEqual(l1.head.num, 33);
118         l1.removHead();
119         Assert.IsTrue(l1.head == null);
120     }
121     [TestMethod]
122     public void TestMethod9_removHead3()
123     {
124         //When the dll is empty
125         DLLList l = new DLLList();
126
127         l.removHead();
128         Assert.AreEqual(l.head, null);
129     }
130     //Remove Tail
131     [TestMethod]
132     public void TestMethod10_removeTail1()
133     {
134         //when there are more than 1 nodes
135         DLLNode p = new DLLNode(33);
136         DLLNode q = new DLLNode(34);
137         DLLList l1 = new DLLList();
138         l1.addToHead(q);
139         l1.addToHead(p);
140
141         l1.removeTail();
142
143         Assert.AreEqual(l1.tail.num, 33);
144     }
145     [TestMethod]
```

```
146     public void TestMethod11_removeTail2()
147     {
148         //when there is only 1 node
149         DLLNode p = new DLLNode(33);
150         DLLList l1 = new DLLList();
151
152         l1.addToHead(p);
153         //Assert.AreEqual(l1.tail.num, 33); //Pre-check
154         l1.removeTail();
155
156         Assert.AreEqual(l1.tail, null);
157     }
158     [TestMethod]
159     public void TestMethod12_removeTail3()
160     {
161         //When the dll is empty
162         DLLList l = new DLLList();
163
164         l.removeTail();
165
166         Assert.IsTrue(l.tail == null);
167     }
168
169     //search
170
171     [TestMethod]
172     public void TestMethod13_search1()
173     {
174         //1 node DLL
175         DLLNode p = new DLLNode(33);
176         DLLList l = new DLLList();
177
178         l.addToHead(p);
179         DLLNode q = l.search(33);
180
181         Assert.AreEqual(p, q);
182     }
183
184
185
186     [TestMethod]
187     public void TestMethod14_search2()
188     {
189         //more than 1 nodes DLL with 1 node meet the requirement
190         DLLNode p = new DLLNode(33);
191         DLLNode q = new DLLNode(34);
192         DLLList l = new DLLList();
193
194         l.addToHead(q);
```

```
195         l.addToHead(p);
196         DLLNode x = l.search(33);
197
198         Assert.AreEqual(x, p);
199     }
200
201     [TestMethod]
202     public void TestMethod15_search3()
203     {
204         //more than 1 nodes DLL with 1 node meet the requirement
205         DLLNode p = new DLLNode(33);
206         DLLNode q = new DLLNode(34);
207         DLLList l = new DLLList();
208
209         l.addToHead(q);
210         l.addToHead(p);
211         DLLNode x = l.search(34);
212
213         Assert.AreEqual(x, q);
214     }
215     [TestMethod]
216     public void TestMethod16_search4()
217     {
218         //not found
219         DLLNode p = new DLLNode(33);
220         DLLNode q = new DLLNode(34);
221         DLLList l = new DLLList();
222
223         l.addToHead(q);
224         l.addToHead(p);
225         DLLNode x = l.search(35);
226
227         Assert.AreNotEqual(x, q);
228         Assert.AreNotEqual(x, p);
229     }
230
231
232     /* [TestMethod]
233     public void TestMethod15_search3()
234     {
235         //more than 1 nodes DLL with more than 1 nodes meet the requirement
236
237         var pointers = new List<int>();
238         var pointers_expected = new List<int>();
239
240         pointers_expected.Add(1);
241         pointers_expected.Add(0);
242
```

```
243         DLLNode p = new DLLNode(33);
244         DLLNode q = new DLLNode(33);
245         DLLList l = new DLLList();
246         l.addToHead(q);
247         l.addToHead(p);
248
249         while q
250
251             int pointer = l.search(33);
252             Assert.AreEqual(1, pointer);
253
254     }
255     */
256     /*
257     public void TestMethod15_search3()
258     {
259         //searched item not in the dll !!!!!
260         DLLNode p = new DLLNode(33);
261         DLLNode q = new DLLNode(34);
262         DLLList l = new DLLList();
263
264         l.addToHead(q);
265         l.addToHead(p);
266         bool pointer = l.search(32);
267
268         Assert.IsTrue(pointer);
269
270     }
271     */
272     //Remove Node
273
274     [TestMethod]
275     public void TestMethod16_removeNode1()
276     {
277         //Not inside !!!!!
278         DLLNode p = new DLLNode(33);
279         DLLNode q = new DLLNode(34);
280         DLLList l = new DLLList();
281
282         l.addToHead(p);
283         l.removeNode(q);
284
285         Assert.AreEqual(l.head, p);
286     }
287     [TestMethod]
288     public void TestMethod17_removeNode2()
289     {
290         //Node inside, 1 Node only
291         DLLNode p = new DLLNode(33);
```

```
292         DLLList l = new DLLList();
293
294         l.addToHead(p);
295         l.removeNode(p);
296
297         Assert.AreEqual(l.head, null);
298         //Assert.AreEqual(p, l);
299     }
300     [TestMethod]
301     public void TestMethod18_removeNode3()
302     {
303         //Node inside, more than 1 Node, head
304         DLLNode p = new DLLNode(33);
305         DLLNode q = new DLLNode(34);
306         DLLList l = new DLLList();
307
308         l.addToHead(q);
309         l.addToHead(p);
310
311         l.removeNode(p);
312
313         Assert.AreEqual(l.head, q);
314     }
315     [TestMethod]
316     public void TestMethod19_removeNode4()
317     {
318         //Node inside, more than 1 Node, tail
319         DLLNode p = new DLLNode(33);
320         DLLNode q = new DLLNode(34);
321         DLLList l = new DLLList();
322
323         l.addToHead(q);
324         l.addToHead(p);
325         l.removeNode(q);
326
327         Assert.AreEqual(l.head, p);
328     }
329
330     //Total
331     [TestMethod]
332     public void TestMethod20_Total1()
333     {
334         //0 Nodes for total
335         DLLList l = new DLLList();
336
337         Assert.AreEqual(l.total(), 0);
338     }
339     [TestMethod]
340     public void TestMethod21_Total2()
```

```
341     {
342         // 1 Node for total
343         DLLNode p = new DLLNode(33);
344         DLLList l = new DLLList();
345         l.addToTail(p);
346
347         Assert.IsTrue(l.total()==33);
348     }
349     [TestMethod]
350     public void TestMethod22_Total3()
351     {
352         // > 1 Nodes for total
353         DLLNode p = new DLLNode(33);
354         DLLNode q = new DLLNode(34);
355         DLLList l = new DLLList();
356         l.addToHead(p);
357         l.addToHead(q);
358
359         Assert.IsTrue(l.total()==67);
360     }
361 }
362 }
```