```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6
 7  namespace DoublyLinkedListWithErrors
 8  {
 9     public class DLList
10      {
11          public DLLNode head; // pointer to the head of the list
12          public DLLNode tail; // pointer to the tail of the list
13         public DLList() { head = null;  tail = null; } // constructor
14          /                                                                ↵
              *---------------------------------------------------------------↵
              ----
15          * The methods below includes several errors. Your  task is to     ↵
              write
16          * unit test to discover these errors. During delivery the tutor   ↵
              may
17          * add or remove errors to adjust the scale of the effort required ↵
              by
18          */
19          public void addToTail(DLLNode p)
20          {
21
22              if (head == null)
23              {
24                  head = p;
25                  tail = p;
26              }
27              else
28              {
29                  tail.next = p;
30                  p.previous = tail;
31                  tail = p;
32              }
33          } // end of addToTail
34
35          public void addToHead(DLLNode p)
36          {
37              if (head == null)
38              {
39                  head = p;
40                  tail = p;
41              }
42              else
43              {
44                  this.head.previous = p;
```

```
45                  p.next = this.head;
46
47                  this.head = p;
48              }
49          } // end of addToHead
50
51          public void removHead()
52          {
53              if (this.head == null) return;
54              if (this.tail == this.head) {
55                  //1 node
56                  this.tail = null;
57                  this.head = null;
58                  return;
59              }
60              //more than 1 nodes
61
62              this.head = this.head.next;
63              this.head.previous = null;
64          } // removeHead
65
66          public void removeTail()
67          {
68              if (this.tail == null) return;
69              //first 1 can be merged into sencond since null is null
70              if (this.head == this.tail)
71              {
72                  this.head = null;
73                  this.tail = null;
74                  return;
75              }
76              this.tail = this.tail.previous;
77              this.tail.next = null;
78              //this.tail.previous.next = null;
79              //this.tail = this.tail.previous;
80          } // remove tail
81
82          /*-------------------------------------------------
83           * Return null if the string does not exist.
84           * ------------------------------------------------*/
85          public DLLNode search(int num)
86          {
87              //if the node is inside, return the node, else return null
88
89              DLLNode q = this.head;
90
91              while (q != null)
92              {
93                  if (q.num == num)
```

```
 94                  {
 95                      return(q);
 96                  }
 97                  q = q.next;
 98              }
 99              return (null);
100
101          } // end of search (return pionter to the node);
102
103          public void removeNode(DLLNode p)
104          { // removing the node p.
105
106
107              bool found = false;
108              DLLNode q = this.head;
109
110              while (q != null)
111              {
112                  if (q.num == p.num)
113                  {
114                      found = true;
115                      break;
116                  }
117                  q = q.next;
118              }
119
120              //not found in 0 nodes dll, 1 node dll, or more nodes dll
121              if (found == false)
122              {
123                  return;
124              }
125
126              //found
127              else if(found == true)
128              {
129                  //1 node
130                  if (this.tail == this.head)
131                  {
132                      //this.tail == this.head && this.tail.num == p.num
133                      this.tail = null;
134                      this.head = null;
135                      return;
136                  }
137                  else if (q.next == null)
138                  {
139                      this.tail = this.tail.previous;
140                      this.tail.next = null;
141                      //q.previous = null;
142                      return;
```

```
143                 }
144                 else if (q.previous == null)
145                 {
146                     this.head = this.head.next;
147                     q.next = null;
148                     this.head.previous = null;
149                     return;
150                 }
151                 p.previous.next = p.next;
152                 p.next.previous = p.previous;
153 
154                 p.next = null;
155                 p.previous = null;
156                 return;
157             }
158         } // end of remove a node
159 
160         public int total()
161         {
162             DLLNode p = head;
163             int tot = 0;
164             while (p != null)
165             {
166                 tot += p.num;
167                 p = p.next;
168             }
169             return (tot);
170         } // end of total
171     } // end of DLList class
172 }
173 
```