```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6
 7  namespace Unit4
 8  {
 9      class RoundNodeUtils
10      {
11          public static void MakeListRound<T>(Node<T> lst)
12          {
13              Node<T> head = lst;
14
15              while (lst.GetNext() != null)
16              {
17                  lst = lst.GetNext();
18              }
19
20              lst.SetNext(head);
21          }
22
23          public static void PrintRoundList<T>(Node<T> lst)
24          {
25
26              Node<T> pos = lst.GetNext();
27
28              Console.Write(lst + "-->");
29
30              while (pos != lst)
31              {
32                  Console.Write(pos + "-->");
33                  pos = pos.GetNext();
34              }
35          }
36
37          public static void DisconnectRoundList<T>(Node<T> lst)
38          {
39              Node<T> pos = lst.GetNext();
40
41              while (pos.GetNext() != lst)
42              {
43                  pos = pos.GetNext();
44              }
45
46              pos.SetNext(null);
47          }
48
49          public static bool IsRoundList<T>(Node<T> head)
```

```
50          {
51              Node<T> pos = head.GetNext();
52
53              while (pos != head && pos != null)
54              {
55                  pos = pos.GetNext();
56              }
57
58              return pos == head;
59          }
60
61          public static int ListLength<T>(Node<T> head)
62          {
63              Node<T> pos = head.GetNext();
64              int cnt = 1;
65
66              while (pos != head)
67              {
68                  pos = pos.GetNext();
69                  cnt++;
70              }
71
72              return cnt;
73          }
74
75          public static int SumList(Node<int> head)
76          {
77              Node<int> pos = head.GetNext();
78
79              int sum = head.GetValue();
80
81              while(pos != head)
82              {
83                  sum += pos.GetValue();
84                  pos = pos.GetNext();
85              }
86
87              return sum;
88          }
89
90          public static Node<T> RemoveHead<T>(Node<T> head)
91          {
92              Node<T> pos = head;
93
94              while(pos.GetNext() != head)
95              {
96                  pos = pos.GetNext();
97              }
98
```

```
 99              pos.SetNext(head.GetNext());
100              return pos.GetNext();
101          }
102
103          public static void RemoveLast<T>(Node<T> head)
104          {
105              Node<T> pos = head;
106
107              while (pos.GetNext().GetNext() != head)
108              {
109                  pos = pos.GetNext();
110              }
111
112              pos.SetNext(head);
113          }
114
115          public static bool IsExist(Node<int> head, int value)
116          {
117              Node<int> pos = head.GetNext();
118
119              bool found = false;
120
121              while (pos != head && !found)
122              {
123                  found = pos.GetValue() == value;
124                  pos = pos.GetNext();
125              }
126
127              return found;
128          }
129
130          public static Node<int> RemoveEven(Node<int> head)
131          {
132              Node<int> last = head;
133              Node<int> pos = head.GetNext();
134
135              while (pos != head)
136              {
137                  if (pos.GetValue() % 2 == 0)
138                  {
139                      last.SetNext(pos.GetNext());
140                  }
141
142                  else
143                      last = pos;
144
145                  pos = pos.GetNext();
146              }
147
```

```
148             if (pos.GetValue() % 2 == 0)
149             {
150                 last.SetNext(pos.GetNext());
151                 return pos.GetNext();
152             }
153 
154             return head;
155         }
156 
157         public static void AddToEven(Node<int> head)
158         {
159             Node<int> last = head;
160             Node<int> pos = head.GetNext();
161 
162             while(pos != head)
163             {
164                 if (pos.GetValue() % 2 == 0)
165                 {
166                     last.SetNext(new Node<int>(pos.GetValue() - 1));
167                     last.GetNext().SetNext(pos);
168                     last = pos;
169                 }
170 
171                 else
172                     last = pos;
173 
174                 pos = pos.GetNext();
175             }
176 
177             if (pos.GetValue() % 2 == 0)
178             {
179                 last.SetNext(new Node<int>(pos.GetValue() - 1));
180                 last.GetNext().SetNext(pos);
181             }
182 
183         }
184 
185         public static Node<T> AddToLoop<T>(Node<T> head, Node<T> new_node)
186         {
187             Node<T> pos = head.GetNext();
188 
189             while (pos.GetNext() != head)
190                 pos = pos.GetNext();
191 
192             pos.SetNext(new_node);
193             new_node.SetNext(head);
194 
195             return new_node; // new_node => new list head
196         }
```

```
197
198         public static void SumNeighbors(Node<int> head)
199         {
200             Node<int> last = head;
201             Node<int> pos = head.GetNext();
202
203             while(pos != head)
204             {
205                 AddToLoop(pos, new Node<int>(last.GetValue() +
                      pos.GetValue()));
206                 last = pos;
207                 pos = pos.GetNext();
208             }
209             AddToLoop(pos, new Node<int>(last.GetValue() + pos.GetValue
                  ()));
210         }
211
212         public static bool HasLoop<T>(Node<T> head)
213         {
214             Node<T> curr = head;
215             Node<T> next = head.GetNext();
216             Node<T> next_next = next.GetNext();
217
218             while (next != null && curr != next_next)
219
220             {
221                 next.SetNext(curr);
222
223                 curr = next;
224                 next = next_next;
225                 if (next_next != null)
226                     next_next = next_next.GetNext();
227             }
228
229             bool foundLoop = curr == next_next;
230
231             return foundLoop;
232         }
233
234         public static void CreateLoopList<T>(Node<T> lst, int n)
235         {
236             Node<T> pos = lst;
237             for (int i = 0; i < n; i++)
238             {
239                 pos = pos.GetNext();
240             }
241
242             while (lst.GetNext() != null)
243                 lst = lst.GetNext();
```

```
244
245             lst.SetNext(pos);
246         }
247
248         public static Node<T> IntersectionPoint<T>(Node<T> rLst)
249         {
250             Node<T> turtle = rLst;
251             Node<T> rabbit = rLst;
252
253             bool found_intersect = true;
254
255             while (rabbit.GetNext().GetNext() != null && found_intersect)
256
257                 for (int i = 0; i < 2; i++)
258                     rabbit = rabbit.GetNext();
259
260                 turtle = turtle.GetNext();
261
262                 found_intersect = rabbit != turtle;
263             }
264
265             if (rabbit == turtle)
266                 return rabbit;
267
268             return null;
269         }
270
271         public static Node<T> CrossSection<T>(Node<T> lst)
272         {
273             Node<T> intersect = IntersectionPoint(lst);
274
275             while (intersect != lst)
276             {
277                 intersect = intersect.GetNext();
278                 lst = lst.GetNext();
279             }
280
281             return intersect;
282         }
283
284         public static void PrintLoopRoundList<T>(Node<T> lst)
285         {
286             Node<T> intersect = IntersectionPoint(lst);
287             while(lst != intersect)
288             {
289                 Console.Write(lst.GetValue() + "-->");
290                 lst = lst.GetNext();
291             }
292
```

```
293             Console.Write("(");
294             Console.Write(lst + "-->");
295             lst = lst.GetNext();
296 
297             while(lst != intersect)
298             {
299                 Console.Write(lst + "-->");
300                 lst = lst.GetNext();
301             }
302 
303             Console.Write(")");
304         }
305     }
306 }
307 
```