

Our Solution(s)

Run Code

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 public class Program {
4     public class DoublyLinkedList {
5         public Node Head;
6         public Node Tail;
7
8         // O(1) time | O(1) space
9         public void SetHead(Node node) {
10             if (Head == null) {
11                 Head = node;
12                 Tail = node;
13                 return;
14             }
15             InsertBefore(Head, node);
16         }
17
18         // O(1) time | O(1) space
19         public void SetTail(Node node) {
20             if (Tail == null) {
21                 SetHead(node);
22                 return;
23             }
24             InsertAfter(Tail, node);
25         }
26
27         // O(1) time | O(1) space
28         public void InsertBefore(Node node, Node nodeToInsert) {
29             if (nodeToInsert == Head && nodeToInsert == Tail) return;
30             Remove(nodeToInsert);
31             nodeToInsert.Prev = node.Prev;
32             nodeToInsert.Next = node;
33             if (node.Prev == null) {
34                 Head = nodeToInsert;
35             } else {
36                 node.Prev.Next = nodeToInsert;
37             }
38             node.Prev = nodeToInsert;
39         }
40
41         // O(1) time | O(1) space
42         public void InsertAfter(Node node, Node nodeToInsert) {
43             if (nodeToInsert == Head && nodeToInsert == Tail) return;
44             Remove(nodeToInsert);
45             nodeToInsert.Prev = node;
46             nodeToInsert.Next = node.Next;
47             if (node.Next == null) {
48                 Tail = nodeToInsert;
49             } else {
50                 node.Next.Prev = nodeToInsert;
51             }
52             node.Next = nodeToInsert;
53         }
54
55         // O(p) time | O(1) space
56         public void InsertAtPosition(int position, Node nodeToInsert) {
57             if (position == 1) {
58                 SetHead(nodeToInsert);
59                 return;
60             }
61         }
62     }
63 }
```

Our Tests

```
1 // Test SetHead
2
3 public class Program {
4     public class DoublyLinkedList {
5         public Node Head;
6         public Node Tail;
7
8         // O(1) time | O(1) space
9         public void SetHead(Node node) {
10             if (Head == null) {
11                 Head = node;
12                 Tail = node;
13                 return;
14             }
15             InsertBefore(Head, node);
16         }
17
18         // O(1) time | O(1) space
19         public void SetTail(Node node) {
20             if (Tail == null) {
21                 SetHead(node);
22                 return;
23             }
24             InsertAfter(Tail, node);
25         }
26
27         // O(1) time | O(1) space
28         public void InsertBefore(Node node, Node nodeToInsert) {
29             if (nodeToInsert == Head && nodeToInsert == Tail) return;
30             Remove(nodeToInsert);
31             nodeToInsert.Prev = node.Prev;
32             nodeToInsert.Next = node;
33             if (node.Prev == null) {
34                 Head = nodeToInsert;
35             } else {
36                 node.Prev.Next = nodeToInsert;
37             }
38             node.Prev = nodeToInsert;
39         }
40
41         // O(1) time | O(1) space
42         public void InsertAfter(Node node, Node nodeToInsert) {
43             if (nodeToInsert == Head && nodeToInsert == Tail) return;
44             Remove(nodeToInsert);
45             nodeToInsert.Prev = node;
46             nodeToInsert.Next = node.Next;
47             if (node.Next == null) {
48                 Tail = nodeToInsert;
49             } else {
50                 node.Next.Prev = nodeToInsert;
51             }
52             node.Next = nodeToInsert;
53         }
54
55         // O(p) time | O(1) space
56         public void InsertAtPosition(int position, Node nodeToInsert) {
57             if (position == 1) {
58                 SetHead(nodeToInsert);
59                 return;
60             }
61         }
62     }
63 }
```

Your Solutions

Run Code

Solution 1 Solution 2 Solution 3

```
1 // Feel free to add new properties and methods to the class.
2 public class Program {
3     public class DoublyLinkedList {
4         public Node Head;
5         public Node Tail;
6
7         public void SetHead(Node node) {
8             // Write your code here.
9         }
10
11        public void SetTail(Node node) {
12            // Write your code here.
13        }
14
15        public void InsertBefore(Node node, Node nodeToInsert) {
16            // Write your code here.
17        }
18
19        public void InsertAfter(Node node, Node nodeToInsert) {
20            // Write your code here.
21        }
22
23        public void InsertAtPosition(int position, Node nodeToInsert) {
24            // Write your code here.
25        }
26
27        public void RemoveNodesWithValue(int value) {
28            // Write your code here.
29        }
30
31        public void Remove(Node node) {
32            // Write your code here.
33        }
34    }
35 }
```

Custom Output

Submit Code

```
1 // Custom Output
2
3 public class Program {
4     public class DoublyLinkedList {
5         public Node Head;
6         public Node Tail;
7
8         // O(1) time | O(1) space
9         public void SetHead(Node node) {
10             if (Head == null) {
11                 Head = node;
12                 Tail = node;
13                 return;
14             }
15             InsertBefore(Head, node);
16         }
17
18         // O(1) time | O(1) space
19         public void SetTail(Node node) {
20             if (Tail == null) {
21                 SetHead(node);
22                 return;
23             }
24             InsertAfter(Tail, node);
25         }
26
27         // O(1) time | O(1) space
28         public void InsertBefore(Node node, Node nodeToInsert) {
29             if (nodeToInsert == Head && nodeToInsert == Tail) return;
30             Remove(nodeToInsert);
31             nodeToInsert.Prev = node.Prev;
32             nodeToInsert.Next = node;
33             if (node.Prev == null) {
34                 Head = nodeToInsert;
35             } else {
36                 node.Prev.Next = nodeToInsert;
37             }
38             node.Prev = nodeToInsert;
39         }
40
41         // O(1) time | O(1) space
42         public void InsertAfter(Node node, Node nodeToInsert) {
43             if (nodeToInsert == Head && nodeToInsert == Tail) return;
44             Remove(nodeToInsert);
45             nodeToInsert.Prev = node;
46             nodeToInsert.Next = node.Next;
47             if (node.Next == null) {
48                 Tail = nodeToInsert;
49             } else {
50                 node.Next.Prev = nodeToInsert;
51             }
52             node.Next = nodeToInsert;
53         }
54
55         // O(p) time | O(1) space
56         public void InsertAtPosition(int position, Node nodeToInsert) {
57             if (position == 1) {
58                 SetHead(nodeToInsert);
59                 return;
60             }
61         }
62     }
63 }
```

```
17 private void reportNewPerson(Person person) { Console.WriteLine(
18     "Program Data: {0}", person);
19     WritePersonToDatabase(person);
20     WritePersonToDatabase(person);
21 }
22
23 private void reportNewPerson(Person person) { Console.WriteLine(
24     "Program Data: {0}", person);
25     WritePersonToDatabase(person);
26     WritePersonToDatabase(person);
27 }
28
29 private void reportNewPerson(Person person) { Console.WriteLine(
30     "Program Data: {0}", person);
31     WritePersonToDatabase(person);
32     WritePersonToDatabase(person);
33 }
```

Run or submit code when you're ready.