

Prompt	Scratchpad	Our Solution(s)	Video Explanation	Run Code	Your Solutions	Run Code
--------	------------	-----------------	-------------------	----------	----------------	----------

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 class Program {
4     static 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             Node node = head;
62             int currentPosition = 1;
63             while (node != null && currentPosition++ != position) node = node.next;
64             if (node != null) {
65                 insertBefore(node, nodeToInsert);
66             } else {
67                 setTail(nodeToInsert);
68             }
69         }
70
71         // O(n) time | O(1) space
72         public void removeNodesWithValue(int value) {
73             Node node = head;
74             while (node != null) {
75                 Node nodeToRemove = node;
76                 node = node.next;
77                 if (nodeToRemove.value == value) remove(nodeToRemove);
78             }
79         }
80     }
81 }
```

Solution 1 Solution 2 Solution 3

```
1 // Feel free to add new properties and methods to the class.
2 class Program {
3     static 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        public boolean containsNodeWithValue(int value) {
36            // Write your code here.
37            return false;
38        }
39    }
40
41    // Do not edit the class below.
42    static class Node {
43        public int value;
44        public Node prev;
45        public Node next;
46
47        public Node(int value) {
48            this.value = value;
49        }
50    }
51 }
52
```

Custom Output Raw Output Submit Code

```
78     }
79 }
80
81 // O(1) time | O(1) space
82 public void remove(Node node) {
83     if (node == head) head = head.next;
84     if (node == tail) tail = tail.prev;
85     removeNodeBindings(node);
86 }
87
88 // O(n) time | O(1) space
89 public boolean containsNodeWithValue(int value) {
90     Node node = head;
91     while (node != null && node.value != value) node = node.next;
92     return node != null;
93 }
94
95 public void removeNodeBindings(Node node) {
96     if (node.prev != null) node.prev.next = node.next;
97     if (node.next != null) node.next.prev = node.prev;
98     node.prev = null;
99     node.next = null;
100 }
101 }
102
103 static class Node {
104     public int value;
```

Run or submit code when you're ready.