

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 Node {
4   constructor(value) {
5     this.value = value;
6     this.prev = null;
7     this.next = null;
8   }
9 }
10
11 class DoublyLinkedList {
12   constructor() {
13     this.head = null;
14     this.tail = null;
15   }
16
17   // O(1) time | O(1) space
18   setHead(node) {
19     if (this.head === null) {
20       this.head = node;
21       this.tail = node;
22       return;
23     }
24     this.insertBefore(this.head, node);
25   }
26
27   // O(1) time | O(1) space
28   setTail(node) {
29     if (this.tail === null) {
30       this.setHead(node);
31       return;
32     }
33     this.insertAfter(this.tail, node);
34   }
35
36   // O(1) time | O(1) space
37   insertBefore(node, nodeToInsert) {
38     if (nodeToInsert === this.head && nodeToInsert === this.tail) return;
39     this.remove(nodeToInsert);
40     nodeToInsert.prev = node.prev;
41     nodeToInsert.next = node;
42     if (node.prev === null) {
43       this.head = nodeToInsert;
44     } else {
45       node.prev.next = nodeToInsert;
46     }
47     node.prev = nodeToInsert;
48   }
49
50   // O(1) time | O(1) space
51   insertAfter(node, nodeToInsert) {
52     if (nodeToInsert === this.head && nodeToInsert === this.tail) return;
53     this.remove(nodeToInsert);
54     nodeToInsert.prev = node;
55     nodeToInsert.next = node.next;
56     if (node.next === null) {
57       this.tail = nodeToInsert;
58     } else {
59       node.next.prev = nodeToInsert;
60     }
61     node.next = nodeToInsert;
62   }
63
64   // O(p) time | O(1) space
65   insertAtPosition(position, nodeToInsert) {
66     if (position === 1) {
67       this.setHead(nodeToInsert);
68       return;
69     }
70     let node = this.head;
71     let currentPosition = 1;
72     while (node !== null && currentPosition++ !== position) node = node.next;
73     if (node !== null) {
74       this.insertBefore(node, nodeToInsert);
75     } else {
76       this.setTail(nodeToInsert);
77     }
78   }
79 }
```

Solution 1 Solution 2 Solution 3

```
1 // This is an input class. Do not edit.
2 class Node {
3   constructor(value) {
4     this.value = value;
5     this.prev = null;
6     this.next = null;
7   }
8 }
9
10 // Feel free to add new properties and methods to the class.
11 class DoublyLinkedList {
12   constructor() {
13     this.head = null;
14     this.tail = null;
15   }
16
17   setHead(node) {
18     // Write your code here.
19   }
20
21   setTail(node) {
22     // Write your code here.
23   }
24
25   insertBefore(node, nodeToInsert) {
26     // Write your code here.
27   }
28
29   insertAfter(node, nodeToInsert) {
30     // Write your code here.
31   }
32
33   insertAtPosition(position, nodeToInsert) {
34     // Write your code here.
35   }
36
37   removeNodesWithValue(value) {
38     // Write your code here.
39   }
40
41   remove(node) {
42     // Write your code here.
43   }
44
45   containsNodeWithValue(value) {
46     // Write your code here.
47   }
48 }
49
50 // Do not edit the line below.
51 exports.Node = Node;
52 exports.DoublyLinkedList = DoublyLinkedList;
53
```

Custom Output Raw Output

Submit Code

```

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

```

Run or submit code when you're ready.