

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

Solution 1

```
54         node = node?.next
55         if nodeToRemove?.value == value {
56             remove(node: nodeToRemove!)
57         }
58     }
59 }
60
61 // O(1) time | O(1) space
62 func insertBefore(node: Node, nodeToInsert: Node) {
63     if nodeToInsert === head, nodeToInsert === tail {
64         return
65     }
66
67     remove(node: nodeToInsert)
68     nodeToInsert.previous = node.previous
69     nodeToInsert.next = node
70
71     if node.previous == nil {
72         head = nodeToInsert
73     } else {
74         node.previous?.next = nodeToInsert
75     }
76
77     node.previous = nodeToInsert
78 }
79
80 // O(1) time | O(1) space
81 func insertAfter(node: Node, nodeToInsert: Node) {
82     if nodeToInsert === head, nodeToInsert === tail {
83         return
84     }
85
86     remove(node: nodeToInsert)
87     nodeToInsert.previous = node
88     nodeToInsert.next = node.next
89
90     if node.next == nil {
91         tail = nodeToInsert
92     } else {
93         node.next?.previous = nodeToInsert
94     }
95
96     node.next = nodeToInsert
97 }
98
99 // O(1) time | O(1) space
100 func setHead(node: Node) {
101     if head == nil {
102         head = node
103         tail = node
104         return
105     }
106
107     insertBefore(node: head!, nodeToInsert: node)
108 }
109
110 // O(1) time | O(1) space
111 func setTail(node: Node) {
112     if tail == nil {
113         setHead(node: node)
114         return
115     }
116
117     insertAfter(node: tail!, nodeToInsert: node)
118 }
119
120 // O(P) time | O(1) space
121 func insertAtPosition(position: Int, nodeToInsert: Node) {
122     if position == 1 {
123         setHead(node: nodeToInsert)
124         return
125     }
126
127     var node = head
128     var currentPosition = 1
129     while node != nil, currentPosition != position {
130         node = node?.next
131         currentPosition = currentPosition + 1
132     }
133
134     if node != nil {
135         insertBefore(node: node!, nodeToInsert: nodeToInsert)
136     } else {
137         setTail(node: nodeToInsert)
138     }
139 }
140
141 func removeNodeBindings(node: Node) {
142     if let previous = node.previous {
143         previous.next = node.next
144     }
```

Solution 1    Solution 2    Solution 3

```
1 class Program {
2     class Node {
3         var value: Int
4         var previous: Node?
5         var next: Node?
6
7         init(value: Int) {
8             self.value = value
9             previous = nil
10            next = nil
11        }
12    }
13
14    class DoublyLinkedList {
15        var head: Node?
16        var tail: Node?
17
18        init() {
19            head = nil
20            tail = nil
21        }
22
23        func containsNodeWithValue(value: Int) -> Bool {
24            // Write your code here.
25            return false
26        }
27
28        func remove(node: Node) {
29            // Write your code here.
30        }
31
32        func removeNodesWithValue(value: Int) {
33            // Write your code here.
34        }
35
36        func insertBefore(node: Node, nodeToInsert: Node) {
37            // Write your code here.
38        }
39
40        func insertAfter(node: Node, nodeToInsert: Node) {
41            // Write your code here.
42        }
43
44        func setHead(node: Node) {
45            // Write your code here.
46        }
47
48        func setTail(node: Node) {
49            // Write your code here.
50        }
51
52        func insertAtPosition(position: Int, nodeToInsert: Node) {
53            // Write your code here.
54        }
55    }
56 }
57
```

Custom Output    Raw Output    Submit Code

```
144 }
145
146     if let next = node.next {
147         next.previous = node.previous
148     }
149
150     node.previous = nil
151     node.next = nil
152 }
153 }
154 }
155
```

**Run or submit code when you're ready.**