

Solution 1

```
1 # Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 class Node:
4     def __init__(self, value):
5         self.value = value
6         self.prev = None
7         self.next = None
8
9
10 class DoublyLinkedList:
11     def __init__(self):
12         self.head = None
13         self.tail = None
14
15     # O(1) time | O(1) space
16     def setHead(self, node):
17         if self.head is None:
18             self.head = node
19             self.tail = node
20             return
21         self.insertBefore(self.head, node)
22
23     # O(1) time | O(1) space
24     def setTail(self, node):
25         if self.tail is None:
26             self.setHead(node)
27             return
28         self.insertAfter(self.tail, node)
29
30     # O(1) time | O(1) space
31     def insertBefore(self, node, nodeToInsert):
32         if nodeToInsert == self.head and nodeToInsert == self.tail:
33             return
34         self.remove(nodeToInsert)
35         nodeToInsert.prev = node.prev
36         nodeToInsert.next = node
37         if node.prev is None:
38             self.head = nodeToInsert
39         else:
40             node.prev.next = nodeToInsert
41         node.prev = nodeToInsert
42
43     # O(1) time | O(1) space
44     def insertAfter(self, node, nodeToInsert):
45         if nodeToInsert == self.head and nodeToInsert == self.tail:
46             return
47         self.remove(nodeToInsert)
48         nodeToInsert.prev = node
49         nodeToInsert.next = node.next
50         if node.next is None:
51             self.tail = nodeToInsert
52         else:
53             node.next.prev = nodeToInsert
54         node.next = nodeToInsert
55
56     # O(p) time | O(1) space
57     def insertAtPosition(self, node, position, nodeToInsert):
58         if position == 1:
59             self.setHead(nodeToInsert)
60             return
61         node = self.head
62         currentPosition = 1
63         while node is not None and currentPosition != position:
64             node = node.next
65             currentPosition += 1
66         if node is not None:
67             self.insertBefore(node, nodeToInsert)
68         else:
69             self.setTail(nodeToInsert)
70
71     # O(n) time | O(1) space
72     def removeNodesWithValue(self, value):
73         node = self.head
74         while node is not None:
75             nodeToRemove = node
76             node = node.next
77             if nodeToRemove.value == value:
78                 self.remove(nodeToRemove)
79
80     # O(1) time | O(1) space
81     def remove(self, node):
82         if node == self.head:
83             self.head = self.head.next
84         if node == self.tail:
85             self.tail = self.tail.prev
86         self.removeNodeBindings(node)
87
88     # O(n) time | O(1) space
89     def containsNodeWithValue(self, value):
90         node = self.head
```

Solution 1

Solution 2

Solution 3

```
1 # This is an input class. Do not edit.
2 class Node:
3     def __init__(self, value):
4         self.value = value
5         self.prev = None
6         self.next = None
7
8
9 # Feel free to add new properties and methods to the class.
10 class DoublyLinkedList:
11     def __init__(self):
12         self.head = None
13         self.tail = None
14
15     def setHead(self, node):
16         # Write your code here.
17         pass
18
19     def setTail(self, node):
20         # Write your code here.
21         pass
22
23     def insertBefore(self, node, nodeToInsert):
24         # Write your code here.
25         pass
26
27     def insertAfter(self, node, nodeToInsert):
28         # Write your code here.
29         pass
30
31     def insertAtPosition(self, position, nodeToInsert):
32         # Write your code here.
33         pass
34
35     def removeNodesWithValue(self, value):
36         # Write your code here.
37         pass
38
39     def remove(self, node):
40         # Write your code here.
41         pass
42
43     def containsNodeWithValue(self, value):
44         # Write your code here.
45         pass
46
```

Custom Output

Raw Output

Submit Code

```
91 while node is not None and node.value != value:
92     node = node.next
93     return node is not None
94
95 def removeNodeBindings(self, node):
96     if node.prev is not None:
97         node.prev.next = node.next
98     if node.next is not None:
99         node.next.prev = node.prev
100     node.prev = None
101     node.next = None
102
```

Run or submit code when you're ready.