Our Solution(s) Run Code

Solution 1

```python
# Copyright © 2020 AlgoExpert, LLC. All rights reserved.

class Node:
    def __init__(self, value):
        self.value = value
        self.prev = None
        self.next = None


class DoublyLinkedList:
    def __init__(self):
        self.head = None
        self.tail = None

    # O(1) time | O(1) space
    def setHead(self, node):
        if self.head is None:
            self.head = node
            self.tail = node
            return
        self.insertBefore(self.head, node)

    # O(1) time | O(1) space
    def setTail(self, node):
        if self.tail is None:
            self.setHead(node)
            return
        self.insertAfter(self.tail, node)

    # O(1) time | O(1) space
    def insertBefore(self, node, nodeToInsert):
        if nodeToInsert == self.head and nodeToInsert == self.tail:
            return
        self.remove(nodeToInsert)
        nodeToInsert.prev = node.prev
        nodeToInsert.next = node
        if node.prev is None:
            self.head = nodeToInsert
        else:
            node.prev.next = nodeToInsert
        node.prev = nodeToInsert

    # O(1) time | O(1) space
    def insertAfter(self, node, nodeToInsert):
        if nodeToInsert == self.head and nodeToInsert == self.tail:
            return
        self.remove(nodeToInsert)
        nodeToInsert.prev = node
        nodeToInsert.next = node.next
        if node.next is None:
            self.tail = nodeToInsert
        else:
            node.next.prev = nodeToInsert
        node.next = nodeToInsert

    # O(p) time | O(1) space
    def insertAtPosition(self, position, nodeToInsert):
        if position == 1:
            self.setHead(nodeToInsert)
            return
        node = self.head
        currentPosition = 1
        while node is not None and currentPosition != position:
            node = node.next
            currentPosition += 1
        if node is not None:
            self.insertBefore(node, nodeToInsert)
        else:
            self.setTail(nodeToInsert)

    # O(n) time | O(1) space
    def removeNodesWithValue(self, value):
        node = self.head
        while node is not None:
            nodeToRemove = node
            node = node.next
            if nodeToRemove.value == value:
                self.remove(nodeToRemove)

    # O(1) time | O(1) space
    def remove(self, node):
        if node == self.head:
            self.head = self.head.next
        if node == self.tail:
            self.tail = self.tail.prev
        self.removeNodeBindings(node)

    # O(n) time | O(1) space
    def containsNodeWithValue(self, value):
        node = self.head
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```python
# This is an input class. Do not edit.
class Node:
    def __init__(self, value):
        self.value = value
        self.prev = None
        self.next = None


# Feel free to add new properties and methods to the class.
class DoublyLinkedList:
    def __init__(self):
        self.head = None
        self.tail = None

    def setHead(self, node):
        # Write your code here.
        pass

    def setTail(self, node):
        # Write your code here.
        pass

    def insertBefore(self, node, nodeToInsert):
        # Write your code here.
        pass

    def insertAfter(self, node, nodeToInsert):
        # Write your code here.
        pass

    def insertAtPosition(self, position, nodeToInsert):
        # Write your code here.
        pass

    def removeNodesWithValue(self, value):
        # Write your code here.
        pass

    def remove(self, node):
        # Write your code here.
        pass

    def containsNodeWithValue(self, value):
        # Write your code here.
        pass
```

Custom Output Raw Output Submit Code

```
        while node is not None and node.value != value:
            node = node.next
        return node is not None

    def removeNodeBindings(self, node):
        if node.prev is not None:
            node.prev.next = node.next
        if node.next is not None:
            node.next.prev = node.prev
        node.prev = None
        node.next = None

```

Run or submit code when you're ready.