Our Solution(s) Run Code

Solution 1 Solution 2

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

class BST:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

    # Average: O(log(n)) time | O(1) space
    # Worst: O(n) time | O(1) space
    def insert(self, value):
        currentNode = self
        while True:
            if value < currentNode.value:
                if currentNode.left is None:
                    currentNode.left = BST(value)
                    break
                else:
                    currentNode = currentNode.left
            else:
                if currentNode.right is None:
                    currentNode.right = BST(value)
                    break
                else:
                    currentNode = currentNode.right
        return self

    # Average: O(log(n)) time | O(1) space
    # Worst: O(n) time | O(1) space
    def contains(self, value):
        currentNode = self
        while currentNode is not None:
            if value < currentNode.value:
                currentNode = currentNode.left
            elif value > currentNode.value:
                currentNode = currentNode.right
            else:
                return True
        return False

    # Average: O(log(n)) time | O(1) space
    # Worst: O(n) time | O(1) space
    def remove(self, value, parentNode=None):
        currentNode = self
        while currentNode is not None:
            if value < currentNode.value:
                parentNode = currentNode
                currentNode = currentNode.left
            elif value > currentNode.value:
                parentNode = currentNode
                currentNode = currentNode.right
            else:
                if currentNode.left is not None and currentNode.right is not None:
                    currentNode.value = currentNode.right.getMinValue()
                    currentNode.right.remove(currentNode.value, currentNode)
                elif parentNode is None:
                    if currentNode.left is not None:
                        currentNode.value = currentNode.left.value
                        currentNode.right = currentNode.left.right
                        currentNode.left = currentNode.left.left
                    elif currentNode.right is not None:
                        currentNode.value = currentNode.right.value
                        currentNode.left = currentNode.right.left
                        currentNode.right = currentNode.right.right
                    else:
                        # This is a single-node tree; do nothing.
                        pass
                elif parentNode.left == currentNode:
                    parentNode.left = currentNode.left if currentNode.left is not
                elif parentNode.right == currentNode:
                    parentNode.right = currentNode.left if currentNode.left is no
                break
        return self

    def getMinValue(self):
        currentNode = self
        while currentNode.left is not None:
            currentNode = currentNode.left
        return currentNode.value

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```python
# Do not edit the class below except for
# the insert, contains, and remove methods.
# Feel free to add new properties and methods
# to the class.
class BST:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

    def insert(self, value):
        # Write your code here.
        # Do not edit the return statement of this method.
        return self

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

    def remove(self, value):
        # Write your code here.
        # Do not edit the return statement of this method.
        return self

```

Custom Output Raw Output Submit Code

Run or submit code when you're ready.