Our Solution(s) Run Code

Solution 1 Solution 2

```go
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

package main

type BST struct {
  Value int

  Left  *BST
  Right *BST
}

// Average: O(log(n)) time | O(log(n)) space
// Worst: O(n) time | O(n) space
func (tree *BST) Insert(value int) *BST {
  if value < tree.Value {
    if tree.Left == nil {
      tree.Left = &BST{Value: value}
    } else {
      tree.Left.Insert(value)
    }
  } else {
    if tree.Right == nil {
      tree.Right = &BST{Value: value}
    } else {
      tree.Right.Insert(value)
    }
  }
  return tree
}

// Average: O(log(n)) time | O(log(n)) space
// Worst: O(n) time | O(n) space
func (tree *BST) Contains(value int) bool {
  if value < tree.Value {
    if tree.Left == nil {
      return false
    } else {
      return tree.Left.Contains(value)
    }
  } else if value > tree.Value {
    if tree.Right == nil {
      return false
    } else {
      return tree.Right.Contains(value)
    }
  }
  return true
}

// Average: O(log(n)) time | O(log(n)) space
// Worst: O(n) time | O(n) space
func (tree *BST) Remove(value int) *BST {
  tree.remove(value, nil)
  return tree
}

func (tree *BST) remove(value int, parent *BST) {
  if value < tree.Value {
    if tree.Left != nil {
      tree.Left.remove(value, tree)
    }
  } else if value > tree.Value {
    if tree.Right != nil {
      tree.Right.remove(value, tree)
    }
  } else {
    if tree.Left != nil && tree.Right != nil {
      tree.Value = tree.Right.getMinValue()
      tree.Right.remove(tree.Value, tree)
    } else if parent == nil {
      if tree.Left != nil {
        tree.Value = tree.Left.Value
        tree.Right = tree.Left.Right
        tree.Left = tree.Left.Left
      } else if tree.Right != nil {
        tree.Value = tree.Right.Value
        tree.Left = tree.Right.Left
        tree.Right = tree.Right.Right
      } else {
        // This is a single-node tree; do nothing.
      }
    } else if parent.Left == tree {
      if tree.Left != nil {
        parent.Left = tree.Left
      } else {
        parent.Left = tree.Right
      }
    } else if parent.Right == tree {
      if tree.Left != nil {
        parent.Right = tree.Left
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```go
package main

// 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.
type BST struct {
  Value int

  Left  *BST
  Right *BST
}

func (tree *BST) Insert(value int) *BST {
  // Write your code here.
  // Do not edit the return statement of this method.
  return tree
}

func (tree *BST) Contains(value int) bool {
  // Write your code here.
  return false
}

func (tree *BST) Remove(value int) *BST {
  // Write your code here.
  // Do not edit the return statement of this method.
  return tree
}

```

Custom Output Raw Output Submit Code

```
      } else {
        parent.Right = tree.Right
      }
    }
  }
}

func (tree *BST) getMinValue() int {
  if tree.Left == nil {
    return tree.Value
  }
  return tree.Left.getMinValue()
}

```

Run or submit code when you're ready.