

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1	Solution 2	Solution 3
<pre>1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved. 2 3 package main 4 5 type BST struct { 6 Value int 7 8 Left *BST 9 Right *BST 10 } 11 12 // Average: O(log(n)) time O(log(n)) space 13 // Worst: O(n) time O(n) space 14 func (tree *BST) Insert(value int) *BST { 15 if value < tree.Value { 16 if tree.Left == nil { 17 tree.Left = &BST{Value: value} 18 } else { 19 tree.Left.Insert(value) 20 } 21 } else { 22 if tree.Right == nil { 23 tree.Right = &BST{Value: value} 24 } else { 25 tree.Right.Insert(value) 26 } 27 } 28 return tree 29 } 30 31 // Average: O(log(n)) time O(log(n)) space 32 // Worst: O(n) time O(n) space 33 func (tree *BST) Contains(value int) bool { 34 if value < tree.Value { 35 if tree.Left == nil { 36 return false 37 } else { 38 return tree.Left.Contains(value) 39 } 40 } else if value > tree.Value { 41 if tree.Right == nil { 42 return false 43 } else { 44 return tree.Right.Contains(value) 45 } 46 } 47 return true 48 } 49 50 // Average: O(log(n)) time O(log(n)) space 51 // Worst: O(n) time O(n) space 52 func (tree *BST) Remove(value int) *BST { 53 tree.remove(value, nil) 54 return tree 55 } 56 57 func (tree *BST) remove(value int, parent *BST) { 58 if value < tree.Value { 59 if tree.Left != nil { 60 tree.Left.remove(value, tree) 61 } 62 } else if value > tree.Value { 63 if tree.Right != nil { 64 tree.Right.remove(value, tree) 65 } 66 } else { 67 if tree.Left != nil && tree.Right != nil { 68 tree.Value = tree.Right.getMinValue() 69 tree.Right.remove(tree.Value, tree) 70 } else if parent == nil { 71 if tree.Left != nil { 72 tree.Value = tree.Left.Value 73 tree.Right = tree.Left.Right 74 tree.Left = tree.Left.Left 75 } else if tree.Right != nil { 76 tree.Value = tree.Right.Value 77 tree.Left = tree.Right.Left 78 tree.Right = tree.Right.Right 79 } else { 80 // This is a single-node tree; do nothing. 81 } 82 } else if parent.Left == tree { 83 if tree.Left != nil { 84 parent.Left = tree.Left 85 } else { 86 parent.Left = tree.Right 87 } 88 } else if parent.Right == tree { 89 if tree.Left != nil { 90 parent.Right = tree.Left</pre>		<pre>1 package main 2 3 // Do not edit the class below except for 4 // the insert, contains, and remove methods. 5 // Feel free to add new properties and methods 6 // to the class. 7 type BST struct { 8 Value int 9 10 Left *BST 11 Right *BST 12 } 13 14 func (tree *BST) Insert(value int) *BST { 15 // Write your code here. 16 // Do not edit the return statement of this method. 17 return tree 18 } 19 20 func (tree *BST) Contains(value int) bool { 21 // Write your code here. 22 return false 23 } 24 25 func (tree *BST) Remove(value int) *BST { 26 // Write your code here. 27 // Do not edit the return statement of this method. 28 return tree 29 } 30</pre>
		<div>Custom Output</div> <div>Raw Output</div> <div>Submit Code</div>

```
91         } else {
92             parent.Right = tree.Right
93         }
94     }
95 }
96 }
97
98 func (tree *BST) getMinValue() int {
99     if tree.Left == nil {
100         return tree.Value
101     }
102     return tree.Left.getMinValue()
103 }
104
```

Run or submit code when you're ready.