

Our Solution(s)

Run Code

Solution 1

Solution 2

```
26         right.insert(value: value)
27     } else {
28         right = BST(value: value)
29     }
30 }
31
32 return self
33 }
34
35 // Average: O(log(n)) time | O(log(n)) space
36 // Worst: O(n) time | O(n) space
37 func contains(value: Int) -> Bool {
38     if value < self.value {
39         if let left = left {
40             return left.contains(value: value)
41         } else {
42             return false
43         }
44     } else if value > self.value {
45         if let right = right {
46             return right.contains(value: value)
47         } else {
48             return false
49         }
50     } else {
51         return true
52     }
53 }
54
55 // Average: O(log(n)) time | O(log(n)) space
56 // Worst: O(n) time | O(n) space
57 func remove(value: Int?, parentNode: BST?) -> BST {
58     if let valueToRemove = value, valueToRemove < self.value {
59         if let left = left {
60             left.remove(value: value, parentNode: self)
61         }
62     } else if let valueToRemove = value, valueToRemove > self.value {
63         if let right = right {
64             right.remove(value: value, parentNode: self)
65         }
66     } else {
67         if let _ = left, let right = right {
68             self.value = right.getMinValue()
69
70             right.remove(value: self.value, parentNode: self)
71         } else if parentNode === nil {
72             if let left = left {
73                 self.value = left.value
74
75                 right = left.right
76
77                 self.left = left.left
78             } else if let right = right {
79                 self.value = right.value
80
81                 left = right.left
82
83                 self.right = right.right
84             } else {
85                 // This is a single-node tree; do nothing.
86             }
87         } else if let parent = parentNode, let parentLeft = parent.left, {
88             if let left = left {
89                 parent.left = left
90             } else {
91                 parent.left = right
92             }
93         }
94     }
95 }
```

Your Solutions

Run Code

Solution 1

Solution 2

Solution 3

```
1 class Program {
2     class BST {
3         var value: Int
4         var left: BST?
5         var right: BST?
6
7         init(value: Int) {
8             self.value = value
9             left = nil
10            right = nil
11        }
12
13        func insert(value: Int) -> BST {
14            // Write your code here.
15            return self
16        }
17
18        func contains(value: Int) -> Bool {
19            // Write your code here.
20            return false
21        }
22
23        func remove(value: Int?, parentNode: BST?) -> BST {
24            // Write your code here.
25            return self
26        }
27    }
28 }
29 }
```

Custom Output

Raw Output

Submit Code

```
92         }
93     } else if let parent = parentNode, let parentRight = parentNode?.right {
94         if let left = left {
95             parent.right = left
96         } else {
97             parent.right = right
98         }
99     }
100 }
101
102 return self
103 }
104
105 func getMinValue() -> Int {
106     if let left = left {
107         return left.getMinValue()
108     } else {
109         return value
110     }
111 }
112 }
113 }
114 }
```

Run or submit code when you're ready.