Our Solution(s) Run Code

Solution 1 Solution 2

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

#include <vector>
using namespace std;

class BST {
public:
  int value;
  BST *left;
  BST *right;

  BST(int val) {
    value = val;
    left = NULL;
    right = NULL;
  }

  // Average: O(log(n)) time | O(log(n)) space
  // Worst: O(n) time | O(n) space
  BST &insert(int val) {
    if (val < value) {
      if (left == NULL) {
        BST *newBST = new BST(val);
        left = newBST;
      } else {
        left->insert(val);
      }
    } else {
      if (right == NULL) {
        BST *newBST = new BST(val);
        right = newBST;
      } else {
        right->insert(val);
      }
    }
    return *this;
  }

  // Average: O(log(n)) time | O(log(n)) space
  // Worst: O(n) time | O(n) space
  bool contains(int val) {
    if (val < value) {
      if (left == NULL) {
        return false;
      } else {
        return left->contains(val);
      }
    } else if (val > value) {
      if (right == NULL) {
        return false;
      } else {
        return right->contains(val);
      }
    } else {
      return true;
    }
  }

  // Average: O(log(n)) time | O(log(n)) space
  // Worst: O(n) time | O(n) space
  BST &remove(int val, BST *parent = NULL) {
    if (val < value) {
      if (left != NULL) {
        left->remove(val, this);
      }
    } else if (val > value) {
      if (right != NULL) {
        right->remove(val, this);
      }
    } else {
      if (left != NULL && right != NULL) {
        value = right->getMinValue();
        right->remove(value, this);
      } else if (parent == NULL) {
        if (left != NULL) {
          value = left->value;
          right = left->right;
          left = left->left;
        } else if (right != NULL) {
          value = right->value;
          left = right->left;
          right = right->right;
        } else {
          // This is a single-node tree; do nothing.
        }
      } else if (parent->left == this) {
        parent->left = left != NULL ? left : right;
      } else if (parent->right == this) {
        parent->right = left != NULL ? left : right;
      }
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```cpp
#include <vector>
using namespace std;

// 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 {
public:
  int value;
  BST *left;
  BST *right;

  BST(int val) {
    value = val;
    left = NULL;
    right = NULL;
  }

  BST &insert(int val) {
    // Write your code here.
    // Do not edit the return statement of this method.
    return *this;
  }

  bool contains(int val) {
    // Write your code here.
    return false;
  }

  BST &remove(int val) {
    // Write your code here.
    // Do not edit the return statement of this method.
    return *this;
  }
};

```

Custom Output Raw Output Submit Code

```
    }
    return *this;
  }

  int getMinValue() {
    if (left == NULL) {
      return value;
    } else {
      return left->getMinValue();
    }
  }
};

```

Run or submit code when you're ready.