File - /Users/moe/Downloads/cs61b/fa20-s1319/lab7/BSTMap.java

```java
import java.util.Iterator;
import java.util.Set;

public class BSTMap<K extends Comparable<K>, V> implements Map61B<K, V> {
    private Node root;
    private int size;

    private class Node {
        private K key;
        private V value;
        private Node left;
        private Node right;

        public Node(K k, V v) {
            key = k;
            value = v;
        }
    }

    public BSTMap() {
        this.clear();
    }
    /** Removes all of the mappings from this map. */
    public void clear() {
        root = null;
        size = 0;
    }

    /* Returns true if this map contains a mapping for the specified key. */
    public boolean containsKey(K key) {
        return get(key) != null;
    }

    /* Returns the value to which the specified key is mapped, or null if this
     * map contains no mapping for the key.
     */
    public V get(K key) {
        return getHelper(key, root);
    }
    private V getHelper(K key, Node p) {
        if (p == null) {
            return null;
        }
        int cmp = key.compareTo(p.key);
        if (cmp > 0) {
            return getHelper(key, p.right);
        } else if (cmp < 0) {
            return getHelper(key, p.left);
        } else {
            return p.value;
        }
    }
    /* Returns the number of key-value mappings in this map. */
```

```
54      public int size() {
55          return size;
56      }
57
58      /* Associates the specified value with the specified key in this map. */
59      public void put(K key, V value) {
60          root = putHelper(key, value, root);
61      }
62
63      private Node putHelper(K key, V value, Node p) {
64          if (p == null) {
65              size++;
66              return new Node (key, value);
67          }
68          int cmp = key.compareTo(p.key);
69          if (cmp > 0) {
70              p.right = putHelper(key, value, p.right);
71          } else if (cmp < 0) {
72              p.left = putHelper(key, value, p.left);
73          } else {
74              p.value = value;
75          }
76          return p;
77      }
78
79      public void printSet() {
80          printInOrder(root);
81          System.out.println();
82      }
83
84      public void printInOrder(Node p) {
85          if (p == null) {
86              return;
87          }
88          printInOrder(p.left);
89          System.out.print("(" + p.key.toString() + ", " + p.value.toString() + ")");
90          printInOrder(p.right);
91      }
92      public V remove(K key) {
93          throw new UnsupportedOperationException();
94      }
95      public V remove(K key, V value) {
96          throw new UnsupportedOperationException();
97      }
98
99      public Set<K> keySet() {
100         throw new UnsupportedOperationException();
101     }
102     public Iterator<K> iterator() {
103         throw new UnsupportedOperationException();
104     }
105
106
```

```
107 }
```