**public boolean duplicateCheck(BinarySearchTree otherTree)**

**{**

**return duplicateCheck(this.root, otherTree.getRoot());**

**}**

**protected boolean duplicateCheck(TreeNode t1, TreeNode t2)**

**{**

**if ((t1 == null) && (t2 == null))**

**return true;**

**else if (countNodes(t1) != countNodes(t2))**

**return false;**

**else**

**{**

**KeyedItem key1 = (KeyedItem) t1.getItem();**

**KeyedItem key2 = (KeyedItem) t2.getItem();**

**String s1 = (String) key1.getKey();**

**String s2 = (String) key2.getKey();**

**if ( ! s1.equals(s2) )**

**return false;**

**else**

**return (duplicateCheck(t1.getLeft(), t2.getLeft()) &&**

**duplicateCheck(t1.getRight(), t2.getRight()));**

**}**

**}**

**public boolean find(KeyedItem newItem)**

**{**

**return find(this.root, newItem) != null;**

**}**

**protected TreeNode find(TreeNode tNode,**

**KeyedItem newItem) {**

**if (tNode == null) {**

**return null;**

**} // end if**

**KeyedItem nodeItem = (KeyedItem)tNode.getItem();**

**// search for the insertion position**

**if (newItem.getKey().compareTo(nodeItem.getKey()) < 0) {**

**return (find(tNode.getLeft(), newItem));**

**}**

**else if (newItem.getKey().compareTo(nodeItem.getKey()) > 0) {**

**return (find(tNode.getRight(), newItem));**

**} // end if**

**else**

**return tNode;**

**} // end find**

**public int size()**

**{**

**return (countNodes(root));**

**}**

**protected int countNodes(TreeNode tNode)**

**{**

**if (tNode == null)**

**return 0;**

**else**

**return 1 + countNodes(tNode.getLeft()) +**

**countNodes(tNode.getRight());**

**}**