Homework Turnin

Account:	6G_06 (rgalanos@fcps.edu)
Section:	6G
Course:	TJHSST APCS 2016-17
Assignment:	07-04
Receipt ID:	e0e9d920ce2c2cf5266f60ce438b4077

Turnin Successful!

The following file(s) were received:

BinarySearchTreeDelete.java (7376 bytes)

```
 1.  //name:    date:
 2. import java.util.Scanner;
 3. /*************************************************************
 4.  Practice with a Binary Search Tree. Uses TreeNode.
 5. Prompt the user for an input string.  Build a BST from the letters.
 6. Display it as a sideways tree.  Prompt the user for a target and delete
 7.  that node, if it exists. Show the updated tree.
 8. *************************************************************/
 9. public class BinarySearchTreeDelete
10. {
11.    private static TreeNode root = null;
12.    public static void main(String[] args)
13.    {
14.       Scanner sc = new Scanner(System.in);
15.       System.out.print("Input string: ");
16.       String s = sc.nextLine();
17.                            //Case 1:      ECSBPWANR
18.                            //Case 2a:     SNTPOR
19.                            //Case 2b:     HBRNVJSZIK
20.                            //Case 3.a.i:  DBNACFSEJHM
21.                            //Case 3.a.ii: NFSAKPXGQ
22.                            //Case 3b:     RNVGPCAE
23.                            //Case root:   N
24.                            //on the handout: HDJAGKFEOLTMNSU
25.       root = buildTree( root, s );
26.       System.out.println( display(root, 0) );
27.       System.out.print("Delete? ");
28.       String target = sc.next();
29.       if( contains( root, target ) )
30.       {
31.          root = delete( root, target );
32.          System.out.println("\n" + target+" deleted.");
33.       }
34.       else
35.          System.out.println("\n" + target+" not found");
36.       System.out.println( display(root, 0) );
37.    }
38.    public static TreeNode buildTree(TreeNode t, String s)
39.    {
40.       for(int k = 0; k < s.length(); k++)
41.          t = insert(t, "" + s.charAt(k));
42.       return t;
43.    }
44.    /*************************
45.    Recursive algorithm to build a BST:  if the node is null, insert the
46.    new node.  Else, if the item is less, set the left node and recur to
47.    the left.  Else, if the item is greater, set the right node and recur
48.    to the right.
49.    *************************/
50.    public static TreeNode insert(TreeNode t, String s)
```

```
51.    {
52.       if(t==null)
53.          return new TreeNode(s);
54.       if(s.compareTo(t.getValue()+"")<0)
55.          t.setLeft(insert(t.getLeft(), s));
56.       else
57.          t.setRight(insert(t.getRight(), s));
58.       return t;
59.    }
60.
61.    private static String display(TreeNode t, int level)
62.    {
63.       String toRet = "";
64.       if(t == null)
65.          return "";
66.       toRet += display(t.getRight(), level + 1); //recurse right
67.       for(int k = 0; k < level; k++)
68.          toRet += "\t";
69.       toRet += t.getValue() + "\n";
70.       toRet += display(t.getLeft(), level + 1); //recurse left
71.       return toRet;
72.    }
73.
74.    public static boolean contains( TreeNode current, String target )
75.    {
76.       while(current != null)
77.       {
78.          int compare = target.compareTo((String)current.getValue());
79.          if( compare == 0 )
80.             return true;
81.          else if(compare<0)
82.             current = current.getLeft();
83.          else if(compare>0)
84.             current = current.getRight();
85.       }
86.       return false;
87.    }
88.    public static TreeNode delete(TreeNode current, String target)
89.    {
90.       TreeNode root = current;
91.       TreeNode parent = null;
92.       while(current !=null)
93.       {
94.          int compare = target.compareTo((String)current.getValue());
95.         // -------->  your code goes here
96.          if(compare<0)
97.          {
98.             parent = current;
99.             current = current.getLeft();
100.         }
101.         else if(compare>0)
102.         {
103.            parent = current;
104.            current = current.getRight();
105.         }
106.         else
107.         {
108.            if(parent == null)
109.            {
110.               if(current.getLeft()==null&&current.getRight()==null)
111.                  return null;
112.               else if(current.getLeft()==null)
113.                  return current.getRight();
114.               else if(current.getRight()==null)
115.                  return current.getLeft();
116.               else if(current.getLeft().getRight()==null)
117.               {
118.                  current.setValue(current.getLeft().getValue());
119.                  current.setLeft(current.getLeft().getLeft());
120.               }
121.               else
122.               {
123.                  current = current.getLeft();
124.                  while(current.getRight().getRight()!=null)
125.                  {
126.                     current = current.getRight();
127.                  }
128.                  root.setValue(current.getRight().getValue());
129.                  if(current.getRight().getLeft()!=null)
130.                     current.setRight(current.getRight().getLeft());
131.                  else
```

```
                    current.setRight(null);
                }
            }
            else if(current.getLeft()==null&&current.getRight()==null)
            {
                if(((Comparable)parent.getLeft().getValue()).compareTo((Comparable)current.getValue())==0)
                    parent.setLeft(null);
                else
                    parent.setRight(null);
            }
            else if(current.getLeft()==null)
            {
                if(((Comparable)parent.getLeft().getValue()).compareTo((Comparable)current.getValue())==0)
                    parent.setLeft(current.getRight());
                else
                    parent.setRight(current.getRight());
            }
            else if(current.getRight()==null)
            {
                if(((Comparable)parent.getLeft().getValue()).compareTo((Comparable)current.getValue())==0)
                    parent.setLeft(current.getLeft());
                else
                    parent.setRight(current.getLeft());
            }
            else
            {
                if(current.getLeft().getRight()!=null)
                {
                    boolean bool = (((Comparable)parent.getRight().getValue()).compareTo((Comparable)current.getVa
                    current = current.getLeft();
                    while(current.getRight().getRight()!=null)
                    {
                        current = current.getRight();
                    }
                    if(current.getRight().getLeft()==null)
                    {
                        if(bool)
                        {
                            parent.getRight().setValue(current.getRight().getValue());
                            current.setRight(null);
                        }
                        else
                        {
                            parent.getLeft().setValue(current.getRight().getValue());
                            current.setRight(null);
                        }
                    }
                    else
                    {
                        if(bool)
                        {
                            parent.getRight().setValue(current.getRight().getValue());
                            current.setRight(current.getRight().getLeft());
                        }
                        else
                        {
                            parent.getLeft().setValue(current.getRight().getValue());
                            current.setRight(current.getRight().getLeft());
                        }

                    }
                }
                else
                {
                    current.setValue(current.getLeft().getValue());
                    current.setLeft(current.getLeft().getLeft());
                }
            }
            current = null;
        }
    }
    return root;  //never reached
  }
}
```