CS 202 Lab/Programming Assignment 5
Due: Before the class on April 1, 2015

In this assignment you are to write several methods for binary search trees. Specifically, do the following:

1. Download the files **IntegerBST.java** from Kodiak, **complete** and **document** the **clone**, **equals**, **height**, **minPath**, **balanced** and **heightBalanced** methods.

```
class IntegerTreeNode
{
     int                 item;
     IntegerTreeNode     left;   // left child
     IntegerTreeNode     right;  // right child
     ...
}
```

```
public class IntegerBST
{
     private IntegerTreeNode   root
     private int               numItems;
     ...

     // Create a deep copy of this tree.
     public Object clone()
     { ... }


     // Determine whether another tree has the
     // same contents as this tree.
     public boolean equals(IntegerBST anotherTree)
     { ... }


     // Find the height of the tree.
     // Return -1 (???) if the tree is empty.
     public int height()
     { ... }
```

```
        // Find the length of the shortest path from the root to any
        // leaf.  Return -1 if the tree is empty.
        public int minPath()
        { ... }


        // Determine whether the tree is perfectly balanced.
        public boolean balanced()
        { ... }


        // Determine whether the tree is height balanced.
        public boolean heightBalanced()
        { ... }

}
```

2. To test your methods, download the program **Lab5.java** from Kodiak to your disk, compile and run the program.

```
/**
 *  This program tests the revised IntegerBST class.
 */

public class Lab5
{
     /**
      *   main method
      */

     public static void main(String[] args)
     {
          IntegerBST[]    trees = new IntegerBST[8];
                          // create an array of IntegerBST objects

          IntegerBST      tree9 = new IntegerBST();

          int[]           nums = {70, 100, 30, 50, 120, 80, 10, 90,
                                  60, 110, 40, 130, 140, 1, 20};
                          // Note that we can create an array in Java
                          // and initialize it  by using an
                          // initializer list

          int[]           nums2 = {50, 40, 30, 20, 10, 15, 25};
```

```
int             i;        // loop control variable
int             n = nums.length;
int             n2 = nums2.length;
boolean         deleted;


for (i = 0; i < 8; ++i)
     trees[i] = new IntegerBST();  // create each tree

for (i = 0; i < n; ++i)
     trees[0].insert(nums[i]);

System.out.println("The height of tree 0 is "
                          + trees[0].height());

System.out.println("The shortest path to a leaf of tree 0"
                          + " is "  + trees[0].minPath());

if (trees[0].balanced())
     System.out.println("Tree 0 is balanced.");
else
     System.out.println("Tree 0 is not balanced.");

if (trees[0].heightBalanced())
     System.out.println("Tree 0 is height balanced.\n");
else
     System.out.println("Tree 0 is not height balanced.\n");

trees[1] = (IntegerBST) trees[0].clone();
trees[2] = (IntegerBST) trees[0].clone();

trees[1].delete(30);

for (i = 0; i < 3; ++i)
{
     System.out.print("The integer tree " + i + " is: ");
     trees[i].traverse();
}

testEquality(0, trees[0], 1, trees[1]);
testEquality(0, trees[0], 2, trees[2]);

trees[3].insert(5);  trees[3].insert(10);
trees[4].insert(5);  trees[4].insert(30);
trees[5].insert(5);  trees[5].insert(10);
                     trees[5].insert(2);

trees[5].delete(2);
trees[6].insert(10);  trees[6].insert(5);
```

```
            for (i = 4; i < 8; ++i)
                testEquality(3, trees[3], i, trees[i]);

            System.out.println();

            for (i = 0; i < n2; ++i)
                tree9.insert(nums2[i]);

            System.out.println("The height of tree 9 is "
                                + tree9.height());

            System.out.println("The shortest path to a leaf of tree 9"
                                + " is " + tree9.minPath());

            if (tree9.balanced())
                System.out.println("Tree 9 is balanced.");
            else
                System.out.println("Tree 9 is not balanced.");

            if (tree9.heightBalanced())
                System.out.println("Tree 9 is height balanced.\n");
            else
                System.out.println("Tree 9 is not height balanced.\n");

            System.out.println();
        }



        /**
         *  This method tests whether two trees in the
         *  trees array contain the same elements.  It calls
         *  the equals method of the IntegerBST class to test
         *  test for equality of two trees.
         */

        static void testEquality(int n1, IntegerBST t1,
                                 int n2, IntegerBST t2)
        {
            System.out.print("Integer tree " + n1);

            if (t1.equals(t2))
                System.out.print(" = ");
            else
                System.out.print(" != ");

            System.out.println("Integer tree " + n2);
        }
}
```