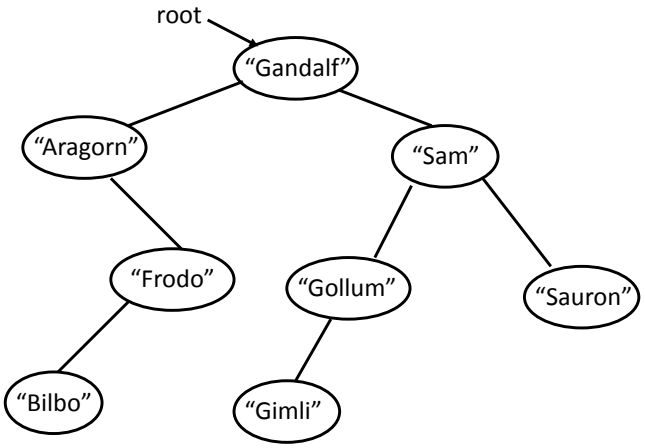
```
public class BinarySearchTree<E extends Comparable>
{

     private static class BinaryTreeNode<E>
     {
          public E data;
          public Node<E> leftChild, rightChild;

          public Node(E data, BinaryTreeNode<E> left, BinaryTreeNode<E> right)
          {
               this.data = data;
               this.leftChild = left;
               this.rightChild = right;
          }
     }


     BinaryTreeNode<E>  root;



     public boolean isInTree(E e)
     {
          return( isInTreeRec( root, e ) );
     }


     public boolean isInTreeRec( v, e )
     {
          if ( v == null )
               return( false );

          System.out.println( v.data.toString() );

          if ( v.data.equals( e ) )
               return( true );

          if ( v.data.compareTo( e ) > 0 )
               return( isInTreeRec( v.leftChild, e ) );

          if ( v.data.compareTo( e ) < 0 )
               return( isInTreeRec( v.rightChild, e ) );

          // System.out.println( v.data.toString() );

     }
}
```

ICS/CSE 22 **Monday, May 23, 2011**

Quiz 7

Instructor: Sandy Irani

1. Suppose we use the class definition on the opposite page to create an instance of `BinarySearchTree<String>`. The `compareTo` method for String compares strings according to alphabetical order. Suppose that the current state of the tree is as pictured in the diagram below.



   (a) Show what is printed out to the console for the method call `isInTree("Frodo")`.

   (b) Now suppose that the print statement is commented out. And the print statement at the end of the method is uncommented. Now what would be printed out to the console for the method call `isInTree("Frodo")`?

2. Consider the binary search tree given above. Show the resulting tree after `add("Pippin")` and `add("Legolas")`. Note that the code for `add` is not given but you should use the algorithm described in class. You should indicate your answer by drawing the new nodes into the tree above.