3. In this question, assume that the `TreeNode` class (whose code is provided in the Appendix) has been modified as shown below to include an additional link to the node’s parent. The following private instance variable and two public methods have been added to the `TreeNode` class to provide this link.

```java
private TreeNode parent;

public TreeNode getParent()
{
    return parent;
}

public void setParent(TreeNode p)
{
    parent = p;
}
```

In the following diagrams, the three-node tree on the left is expanded on the right, showing how parent and child nodes are linked.
Consider the following `Tree` class that maintains and processes a binary search tree whose nodes are objects of the modified `TreeNode` class. You may assume that the tree contains `Comparable` objects and that no two objects stored in the tree are equal. You will implement two of the private methods of the `Tree` class.

```java
public class Tree {
    private TreeNode root;

    // constructs a new, empty tree
    public Tree() {
        root = null;
    }

    // returns the node containing the smallest value in the tree
    // rooted at t when t is not null; otherwise, returns null;
    // the method runs in time O(h), where h is the height
    // of the tree rooted at t
    private TreeNode minNode(TreeNode t) {
        /* implementation not shown */
    }

    // returns the node containing the largest value in the tree
    // rooted at t when t is not null; otherwise, returns null;
    // the method runs in time O(h), where h is the height
    // of the tree rooted at t
    private TreeNode maxNode(TreeNode t) {
        /* implementation not shown */
    }

    // returns true if all nodes in this tree have correct parent links;
    // otherwise, returns false;
    // each node should be the parent of its left and right children;
    // the root node's parent should be null
    private boolean verifyParentLinks() {
        /* to be implemented in part (a) */
    }

    // returns the node which is the successor of t in this tree;
    // returns null if t contains the maximum value in this tree;
    // the method runs in time O(h), where h is the height
    // of the tree rooted at t
    // precondition: t is a node in this tree;
    // all nodes in this tree have correct parent links
    private TreeNode successor(TreeNode t) {
        /* to be implemented in part (b) */
    }
}
```

Copyright © 2005 by College Entrance Examination Board. All rights reserved. Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for AP students and parents).

GO ON TO THE NEXT PAGE.
2005 AP® COMPUTER SCIENCE AB FREE-RESPONSE QUESTIONS

(a) Write the Tree method verifyParentLinks, which tests the structure of the tree. The method traverses the tree and checks whether the parent link in each node correctly references that node's parent. The parent link for the root node should be null.

In writing verifyParentLinks, you may write and use a helper method.

Complete method verifyParentLinks below.

```java
// returns true if all nodes in this tree have correct parent links;
// otherwise, returns false;
// each node should be the parent of its left and right children;
// the root node's parent should be null
private boolean verifyParentLinks()
```

(b) Write the Tree method successor. The successor of a given tree node is defined as the next node that would be visited after the given node during an in-order traversal of the tree. For example, in the first two trees drawn below, the shaded node is the successor of the node containing 50. In the third tree, the node containing 50 is the largest node in the tree and has no successor.

In writing successor, you may use any of the accessible methods in the Tree and modified TreeNode classes.

Complete method successor below.

```java
// returns the node which is the successor of t in this tree;
// returns null if t contains the maximum value in this tree;
// the method runs in time O(h), where h is the height
// of the tree rooted at t
// precondition: t is a node in this tree;
// all nodes in this tree have correct parent links
private TreeNode successor(TreeNode t)
```