 Answers to review questions from Chapter 16

1. What two conditions must be satisfied for a collection of nodes to be a tree?

**• As long as the tree contains any nodes at all, there is a specific node called the *root* that forms the top of a hierarchy.**

**• Every other node is connected to the root by a unique line of descent.**

2. Give at least four real‑world examples that involve tree structures.

**The examples cited in the chapter are family trees, game trees, biological classifications, organizational charts, and directory hierarchies.**

3. Define the terms *parent, child, ancestor, descendant,* and *sibling* as they apply to trees.

**Given a node other than the root, the *parent* of the node is the node to which it connects on the unique path back to the root. The *children* of a node are the nodes for which it is the parent. An *ancestor* of a node is any node in the chain of parents that connect it to the root. A *descendant* of a note is any node for which the original node is an ancestor. Two nodes that share the same parent are called *siblings.***

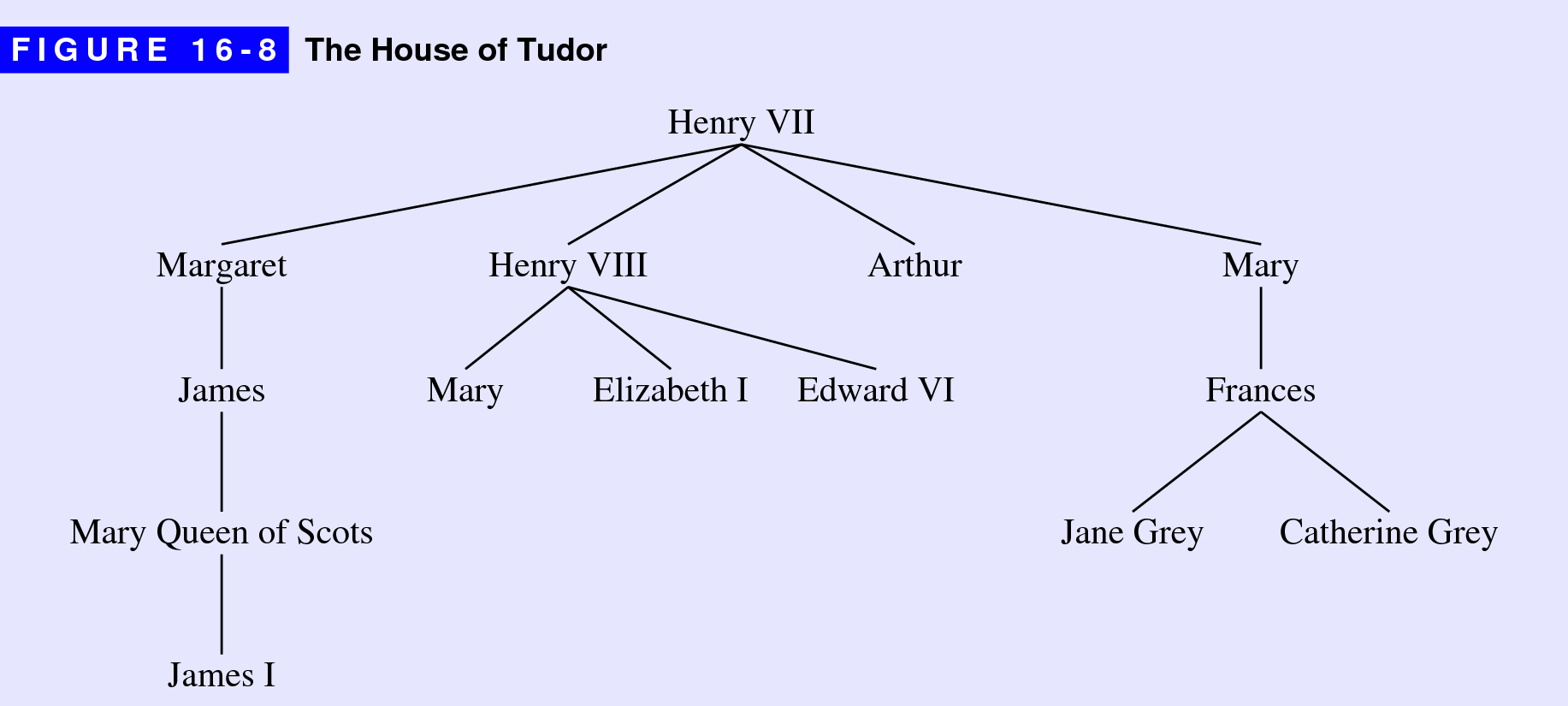
4. The family tree for the House of Tudor, which ruled England in Shakespeare’s time, is shown in Figure 16‑8. Identify the root, leaf, and interior nodes. What is the height of this tree?

**Root: Henry VII**

**Leaf nodes: James I, Mary (daughter of Henry VIII), Elizabeth I, Edward VI, Arthur, Jane Grey, Catherine Grey**

**Interior nodes: Margaret, James, Mary Queen of Scots, Henry VIII, Mary (daughter of Henry VII), Frances**

**The height of the tree is 4, because that is the length of the longest path, which runs from Henry VII to James I.**



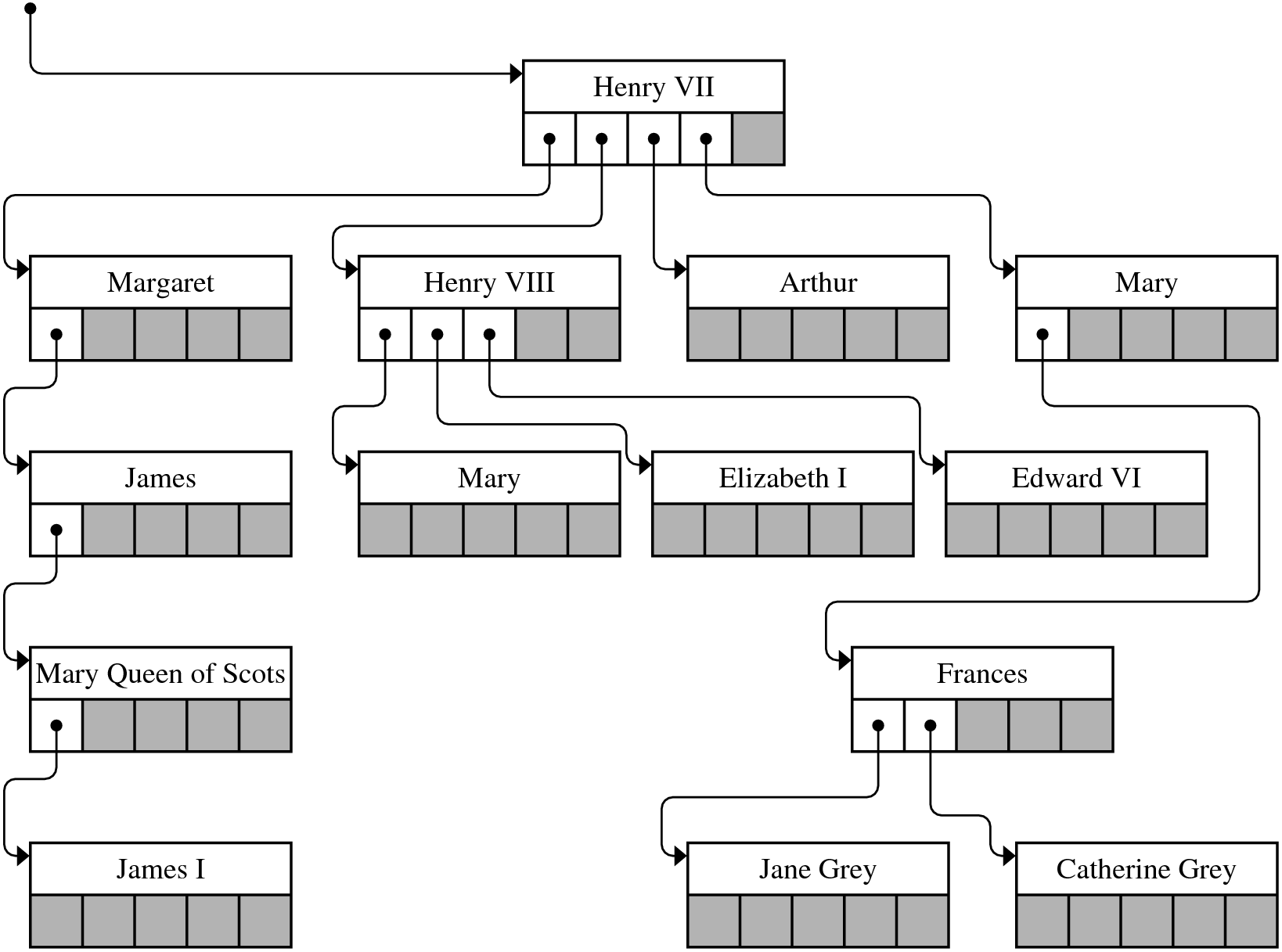
5. What is it about trees that makes them recursive?

**Trees are recursive because each node has children which are themselves trees. The recursion is perhaps clearer in the following mutually recursive definition of trees and nodes:**

**• A *tree* is a pointer to a node.**

**• A *node* is a structure that contain trees.**

6. Diagram the internal structure of the tree shown in Figure 16‑8 when it is represented using the type **FamilyTreeNode**.



7. What is the defining property of a binary search tree?

**1. Every node contains—possibly in addition to other data—a special value called a *key* that defines the order of the nodes.**

**2. Key values are *unique,* in the sense that no key can appear more than once in the tree.**

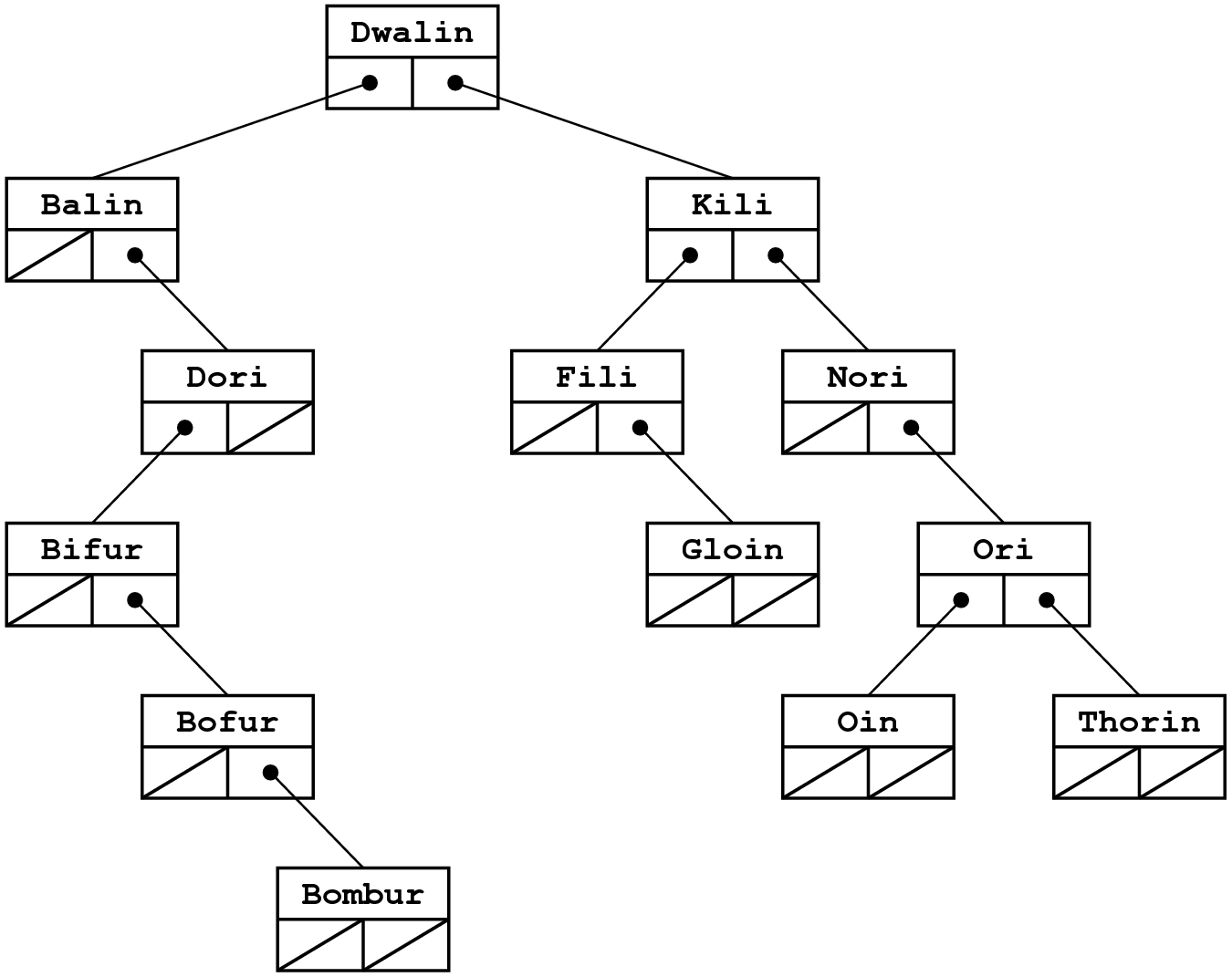
**3. At every node in the tree, the key value must be greater than all the keys in the subtree rooted at its left child and less than all the keys in the subtree rooted at its right child.**

8. Why are different type declarations used for the first argument in **findNode** and **insertNode**?

**The argument to insertNode is passed by reference because the code needs to be able to change that value.**

The next several questions all refer back to this one by number

9. In *The Hobbit* by J. R. R. Tolkien, 13 dwarves arrive at the house of Bilbo Baggins in the following order: **Dwalin**, **Balin**, **Kili**, **Fili**, **Dori**, **Nori**, **Ori**, **Oin**, **Gloin**, **Bifur**, **Bofur**, **Bombur**, and **Thorin**. Diagram the binary search tree that results from inserting these names into an empty tree.



10. Given the tree you created to answer the preceding question, what key comparisons are made if you call **findNode** on the name **Bombur**?

**"Bombur" < "Dwalin"**

**"Bombur" > "Balin"**

**"Bombur" < "Dori"**

**"Bombur" > "Bifur"**

**"Bombur" > "Bofur"**

**"Bombur" = "Bombur"**

11. Write down the preorder, inorder, and postorder traversals of the binary search tree you created for question 9.

**Preorder: Dwalin, Balin, Dori, Bifur, Bofur, Bombur, Kili, Fili, Gloin, Nori, Ori, Oin, Thorin**

**Inorder: Balin, Bifur, Bofur, Bombur, Dori, Dwalin, Fili, Gloin, Kili, Nori, Oin, Ori, Thorin**

**Postorder: Bombur, Bofur, Bifur, Dori, Balin, Gloin, Fili, Oin, Thorin, Ori, Nori, Kili, Dwalin**

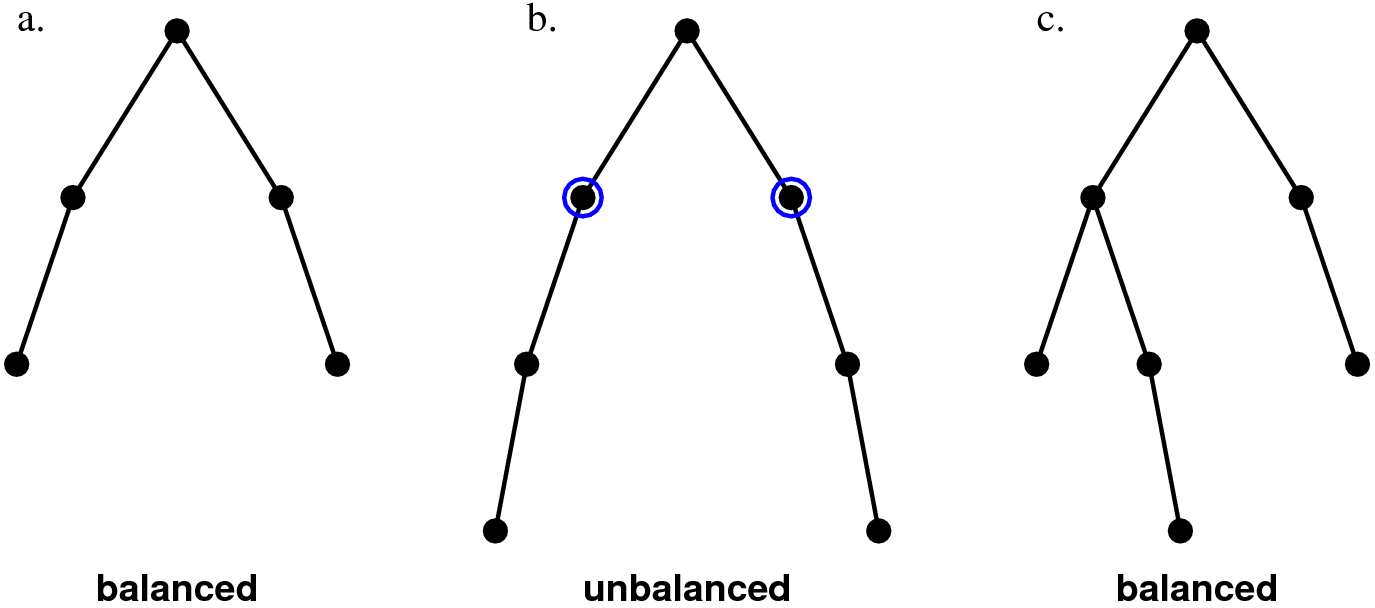
12. One of the three standard traversal orders—preorder, inorder, or postorder—does not depend on the order in which the nodes are inserted into the tree. Which one is it?

**The inorder traversal.**

13. What does it mean for a binary tree to be balanced?

**A binary tree is balanced if every subtree is balanced and the height of the left and right subtrees differ by no more than one.**

14. For each of the following tree structures, indicate whether the tree is balanced:



For any tree structure that is out of balance, indicate which nodes are out of balance.

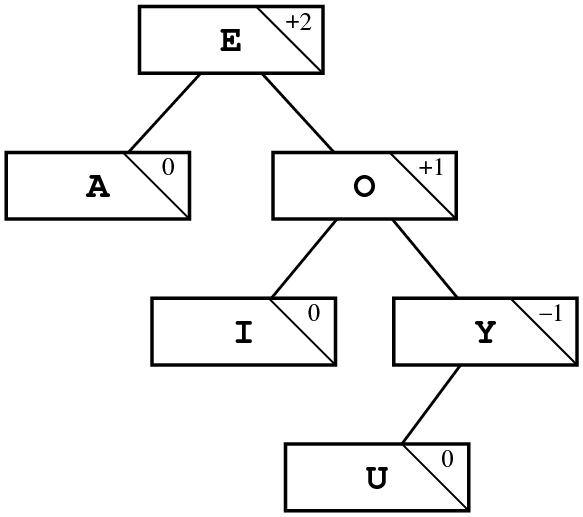
15. True or false: If a binary search tree becomes unbalanced, the algorithms used in the functions **findNode** and **insertNode** will fail to work correctly.

**These functions will continue to work correctly but will run more slowly.**

16. How do you calculate the balance factor of a node?

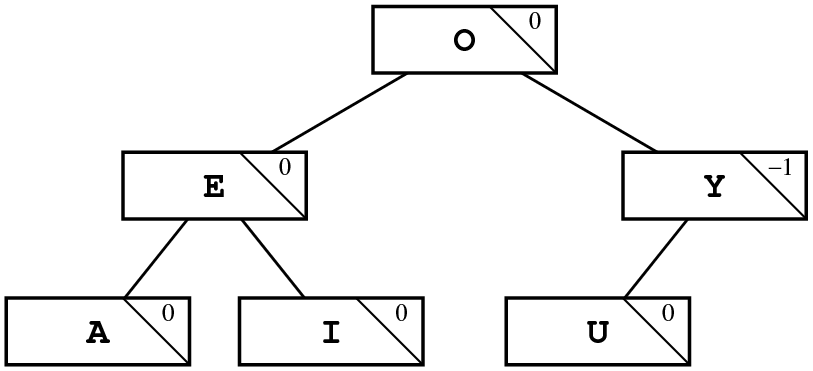
**The balance factor of a node is the height of the right subtree minus the height of the left subtree.**

17. Fill in the balance factors for each node in the following binary search tree:



18. If you use the AVL balancing strategy, what rotation operation must you apply to the tree in the preceding question to restore its balanced configuration? What is the structure of the resulting tree, including the updated balance factors?

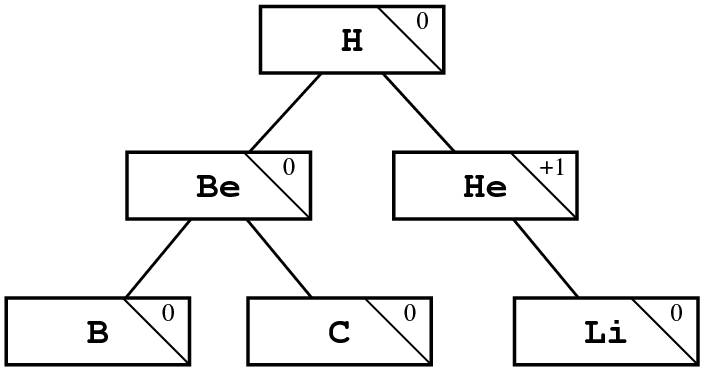
**You need to perform a single left rotation, which produces the following tree:**



19. True or false: When you insert a new node into a balanced binary tree, you can always correct any resulting imbalance by performing one operation, which will be either a single or a double rotation.

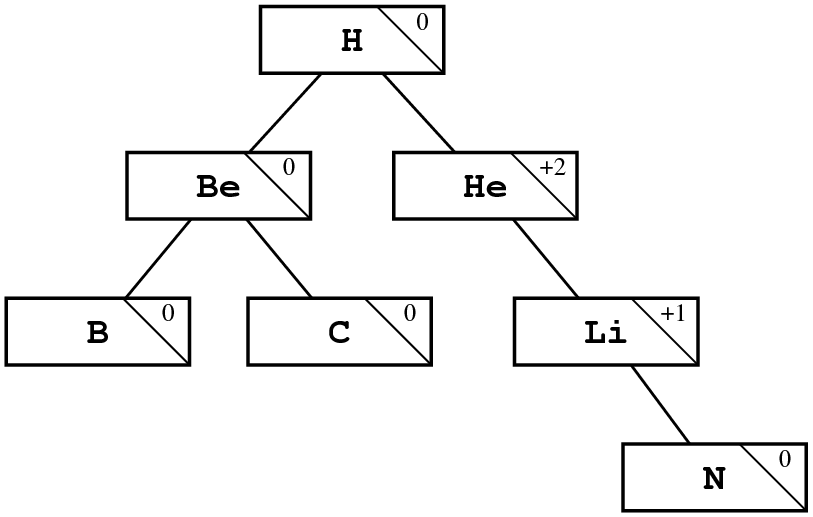
**True.**

20. As shown in the section on “Visualizing the AVL idea,” inserting the symbols for the first six elements into an AVL tree results in the following configuration:

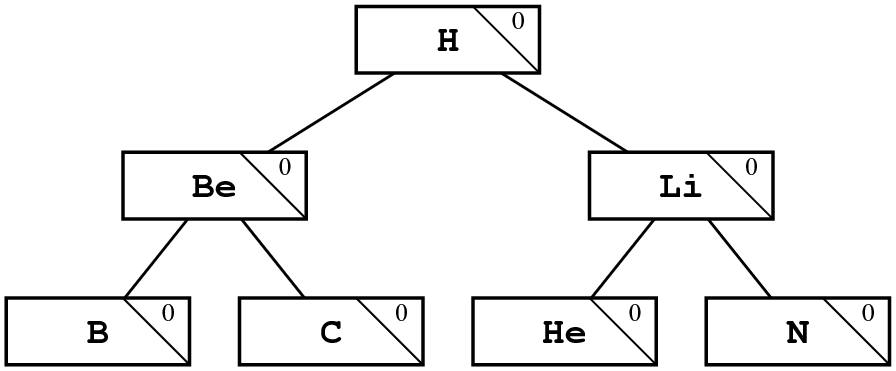


Show what happens to the tree as you add the next six element symbols:

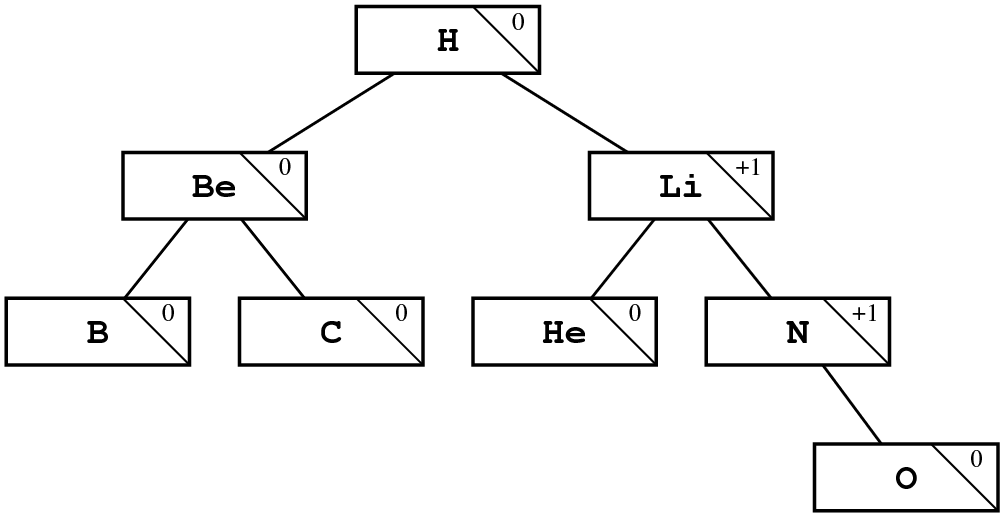
**Step 1: Add** **N (Nitrogen):**



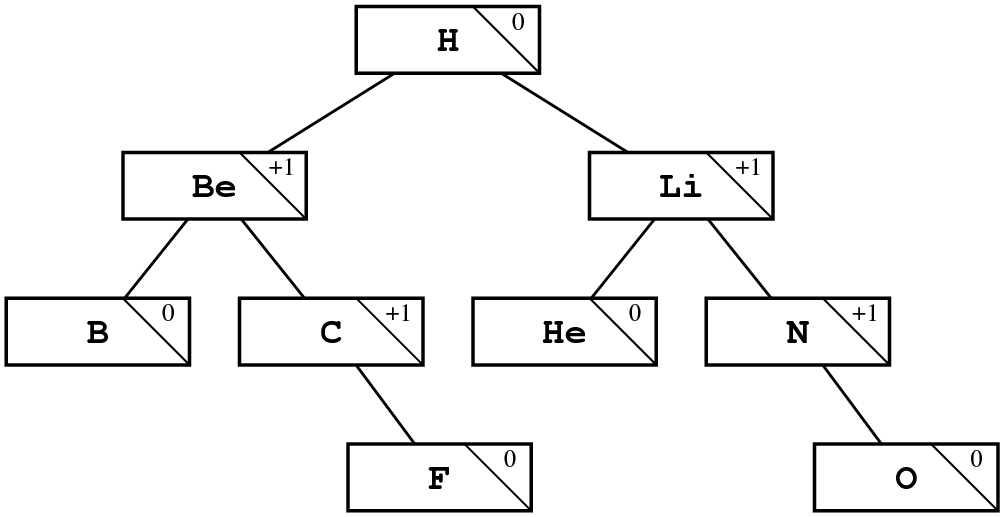
**From which a single left rotation around the He‑Li axis produces the balanced tree**



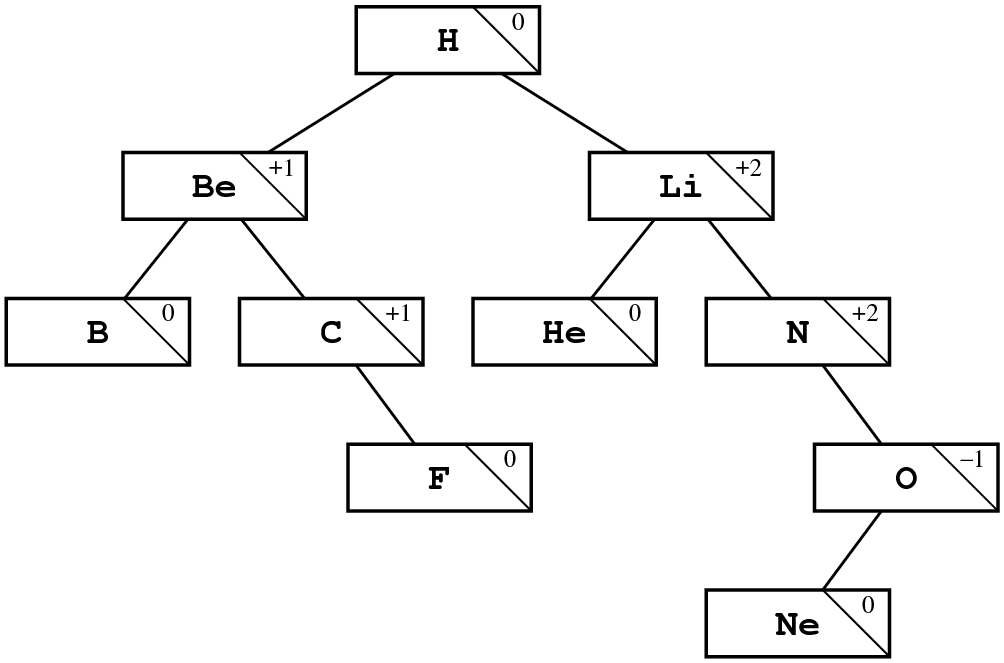
**Step 2: Add O (Oxygen):**



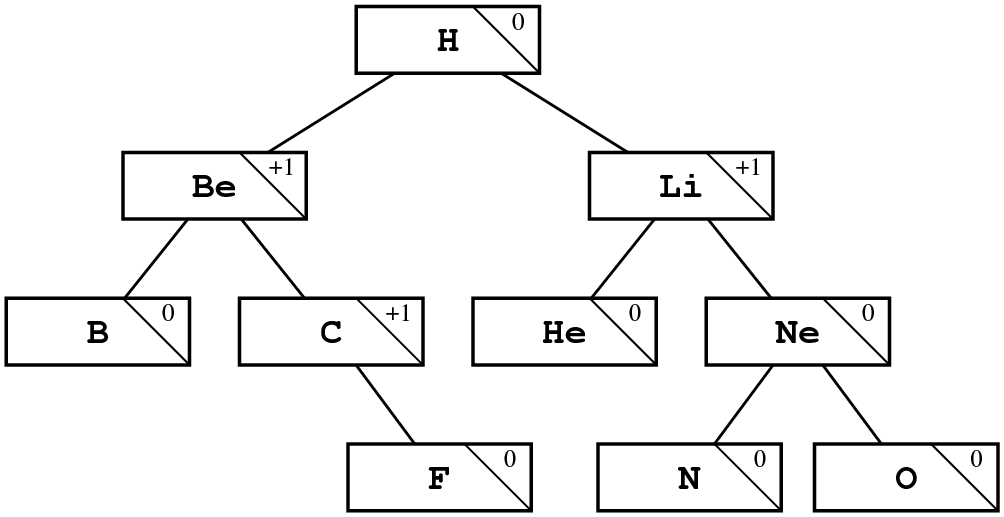
**Step 3: Add F (Fluorine):**



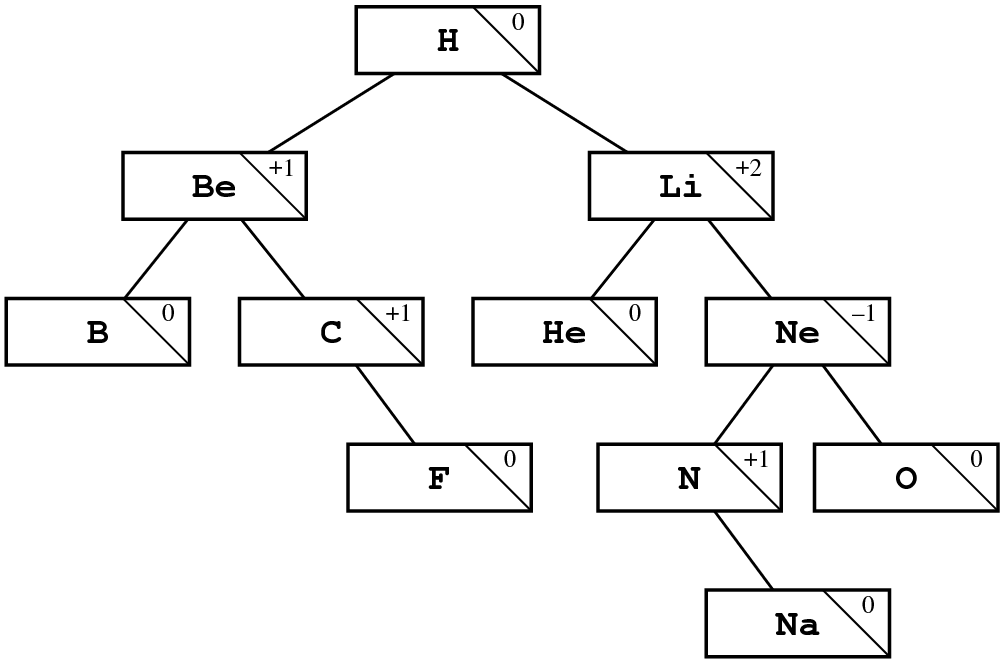
**Step 4: Add Ne (Neon):**



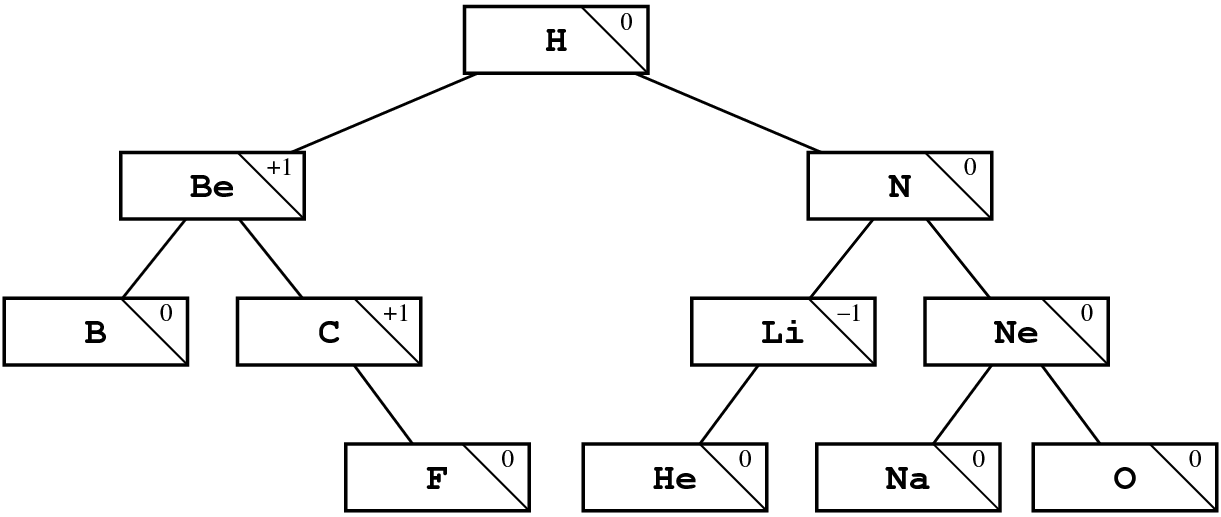
**From which a double rotation around the N‑O axis produces the balanced tree**



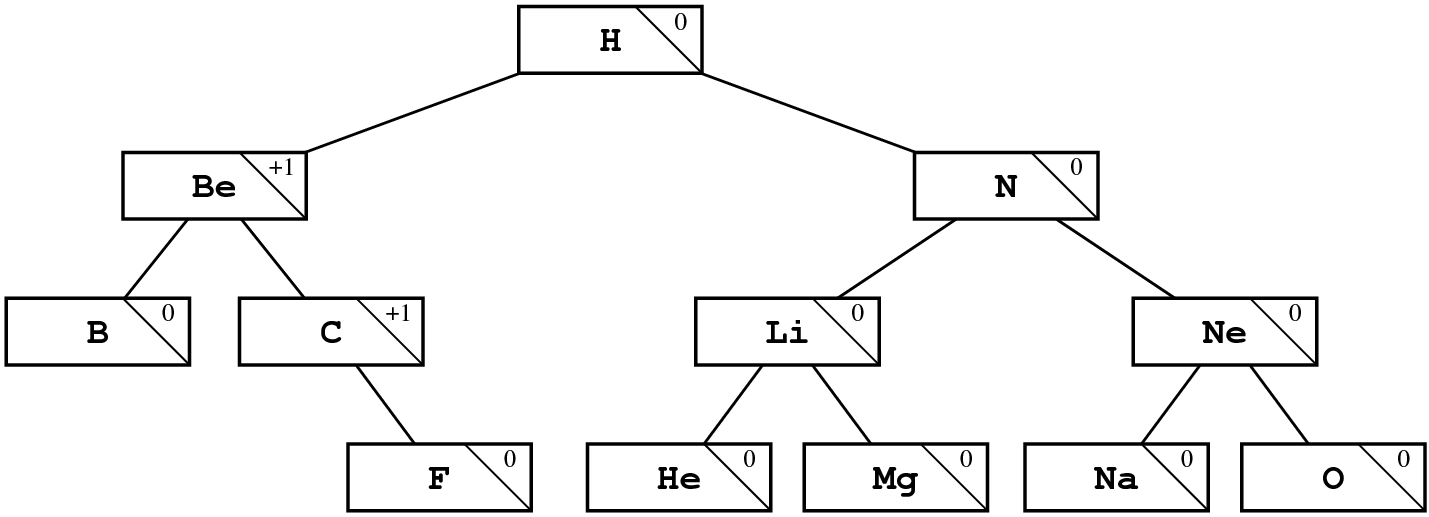
**Step 5: Add Na (Sodium):**



**From which a double rotation around the Li‑N axis produces the balanced tree**



**Step 6: Add Mg (Magnesium):**



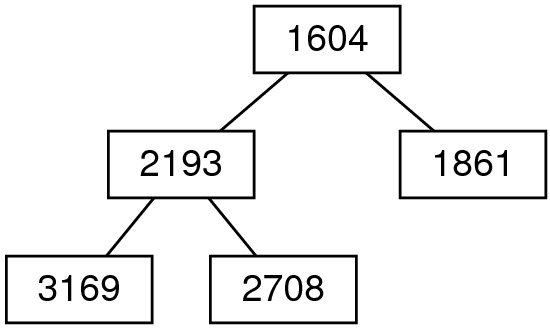
21. Describe in detail what happens during a call to **insertNode**.

**If the tree is NULL, allocate a new node and store it in the parameter variable t, which is passed by reference. Otherwise, check to see whether the node already exists. If it doesn’t, all you have to do is call insertNode recursively on the left or right subtree depending on whether the key comes before or after the one in the current node.**

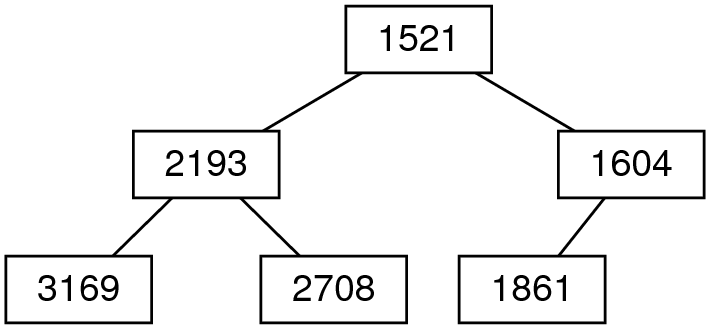
22. What strategy does the text suggest to avoid having a binary search tree become disconnected if you remove an interior node?

**The text replaces the deleted node with the rightmost node in the left subtree.**

23. Suppose that you are working with a partially ordered tree that contains the following data:



Show the state of the partially ordered tree after inserting a node with the key 1521.



24. What is the relationship between heaps and partially ordered trees?

**A *heap* is an array‑based implementation of a partially ordered tree.**