```
    outFile  << " " << tree->info.count;
    outFile << endl;
    Print(tree->right, outFile);
  }
}

void ListType::PrintList(ofstream& outFile)
{
  Print(root, outFile);
}
```

We leave the rest of the problem of creating the index as a programming assignment.

Testing As a test plan for this program, we can take a text file, manually calculate the words and frequencies, and run the program to check the answers. On the Web, the file `Frequency.out` contains the result of running the program on the file `Words.cpp`, using the file `History.in` as input data.

Summary

In this chapter, we saw how we can use a binary tree to structure sorted information so as to reduce the search time for any particular element. For applications requiring direct access to the elements in a sorted structure, the binary search tree is a very useful data type. If the tree is balanced, we can access any node in the tree with an $O(\log_2 N)$ operation. The binary search tree combines the advantages of quick random access (like a binary search on a linear list) with the flexibility of a linked structure.

We also saw that we can implement the tree operations very elegantly and concisely using recursion. This result makes sense, because a binary tree is itself a "recursive" structure: Any node in the tree is the root of another binary tree. Each time we move down a level in the tree, taking either the right or the left path from a node, we cut the size of the (current) tree in half, a clear case of the smaller-caller. We also saw cases of iteration that replaced recursion (`InsertItem` and `DeleteItem`).

Exercises

1. a. What does the level of a binary search tree mean in relation to its searching efficiency?
   b. What is the maximum number of levels that a binary search tree with 100 nodes can have?
   c. What is the minimum number of levels that a binary search tree with 100 nodes can have?
2. Which of these formulas gives the maximum total number of nodes in a tree that has N levels? (Remember that the root is Level 0.)

   a. $N^2 - 1$ b. 2^N c. $2^N - 1$ d. 2^{N+1}

3. Which of these formulas gives the maximum number of nodes in the *N*th level of a binary tree?

   a. N^2 b. 2^N c. 2^{N+1} d. $2^N - 1$

4. How many ancestors does a node in the *N*th level of a binary search tree have?

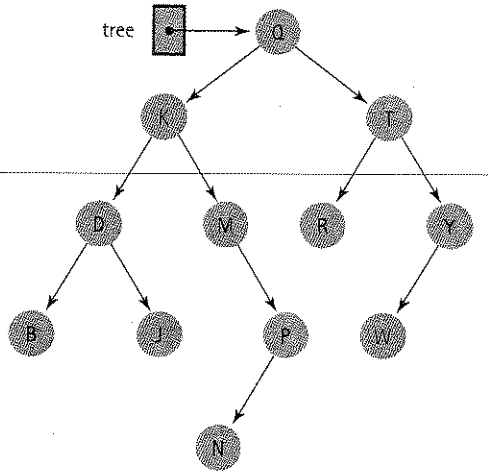
5. a. How many different *binary trees* can be made from three nodes that contain the key values 1, 2, and 3?

   b. How many different *binary search trees* can be made from three nodes that contain the key values 1, 2, and 3?

6. Draw all possible binary trees that have four leaves where all nonleaf nodes have two children.

7. The `TreeType` class used a queue as an auxiliary storage structure for iterating through the elements in the tree. Discuss the relative merits of using a dynamically allocated array-based queue versus a dynamically allocated linked queue.

Answer the questions in Exercises 8–10 independently, using the following tree.



8. a. What are the ancestors of node P?

   b. What are the descendants of node K?

   c. What is the maximum possible number of nodes in the tree at the level of node W?

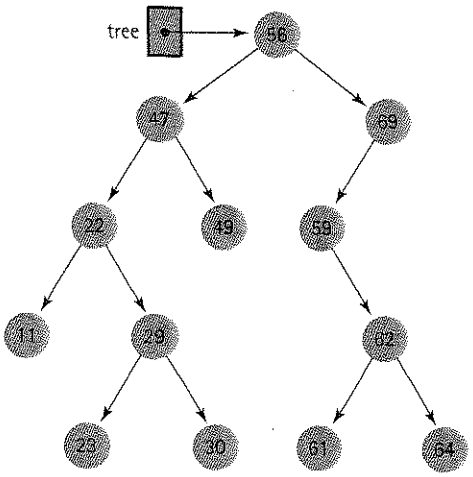
   d. What is the maximum possible number of nodes in the tree at the level of node N?

   e. Insert node O. How many nodes would be in the tree if it were completely full down to and including the level of node O?

9. Show what the tree would look like after each of the following changes. (Use the original tree to answer each part.)
   a. Add node C.
   b. Add node Z.
   c. Add node X.
   d. Delete node M.
   e. Delete node Q.
   f. Delete node R.
10. Show the order in which the nodes in the tree are processed by
   a. an inorder traversal of the tree.
   b. a postorder traversal of the tree.
   c. a preorder traversal of the tree.
11. Draw the binary search tree whose elements are inserted in the following order:

   50 72 96 94 107 26 12 11 9 2 10 25 51 16 17 95

Exercises 12–16 use the following tree.



12. a. What is the height of the tree?
   b. What nodes are on Level 3?
   c. Which levels have the maximum number of nodes that they could contain?
   d. What is the maximum height of a binary search tree containing these nodes? Draw such a tree.
   e. What is the minimum height of a binary search tree containing these nodes? Draw such a tree.
13. a. Trace the path that would be followed in searching for a node containing 61.

b. Trace the path that would be followed in searching for a node containing 28.

14. Show the order in which the nodes in the tree are processed by
    a. an inorder traversal of the tree.
    b. a postorder traversal of the tree.
    c. a preorder traversal of the tree.
15. Show how the tree would look after the deletion of 29, 59, and 47.
16. Show how the (original) tree would look after the insertion of nodes containing 63, 77, 76, 48, 9, and 10 (in that order).
17. True or false?
    a. Invoking the delete function in this chapter might create a tree with more levels than the original tree had.
    b. A preorder traversal processes the nodes in a tree in the exact reverse order that a postorder traversal processes them.
    c. An inorder traversal always processes the elements of a tree in the same order, regardless of the order in which the elements were inserted.
    d. A preorder traversal always processes the elements of a tree in the same order, regardless of the order in which the elements were inserted.
18. If you wanted to traverse a tree, writing all the elements to a file, and later (the next time you ran the program) rebuild the tree by reading and inserting, would an inorder traversal be appropriate? Why or why not?
19. a. One hundred integer elements are chosen at random and inserted into a sorted linked list and a binary search tree. Describe the efficiency of searching for an element in each structure, in terms of Big-O notation.
    b. One hundred integer elements are inserted in order, from smallest to largest, into a sorted linked list and a binary search tree. Describe the efficiency of searching for an element in each structure, in terms of Big-O notation.
20. The key of each node in a binary search tree is a short character string.
    a. Show how such a tree would look after the following words were inserted (in the order indicated):

       monkey canary donkey deer zebra yak walrus vulture penguin quail

    b. Show how the tree would look if the same words were inserted in this order:

       quail walrus donkey deer monkey vulture yak penguin zebra canary

    c. Show how the tree would look if the same words were inserted in this order:

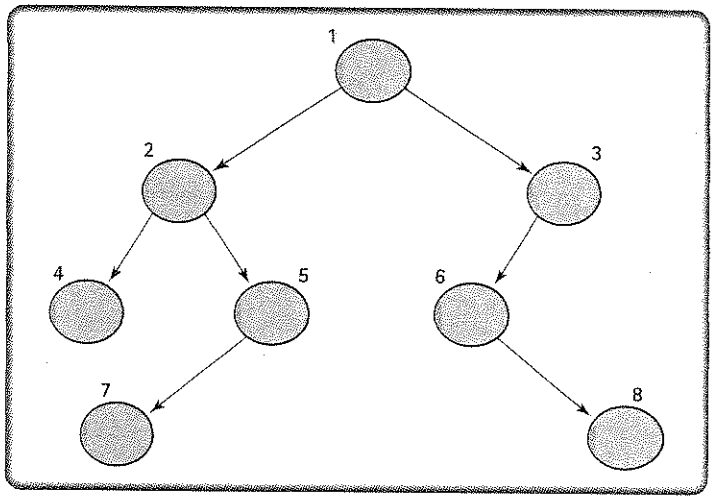
       zebra yak walrus vulture quail penguin monkey donkey deer canary

21. Write a function called `PtrToSuccessor` that finds a node with the smallest key value in a tree, unlinks it from the tree, and returns a pointer to the unlinked node.

22. Modify the `DeleteNode` function so that it uses the immediate successor (rather than the predecessor) of the value to be deleted in the case of deleting a node with two children. You should call the function `PtrToSuccessor` that you wrote in Exercise 21.
23. Use the Three-Question Method to verify the recursive function `Insert`.
24. Use the Three-Question Method to verify the recursive function `Delete`.
25. Write `IsFull` and `IsEmpty` for the iterative version of class `TreeType`.
26. Add a `TreeType` member function `Ancestors` that prints the ancestors of a given node whose `info` member contains `value`. Do not print `value`.
    a. Write the declaration.
    b. Write the iterative implementation.
27. Write a recursive version of the function `Ancestors` described in Exercise 26.
28. Write a recursive version of `Ancestors` (see Exercise 27) that prints out the ancestors in reverse order (first the parent, then the grandparent, and so on).
29. Add a Boolean member function `IsBST` to the class `TreeType` that determines whether a binary tree is a binary search tree.
    a. Write the declaration of the function `IsBST`. Include adequate comments.
    b. Write a recursive implementation of this function.
30. Extend the Binary Search Tree ADT to include the member function `LeafCount` that returns the number of leaf nodes in the tree.
31. Extend the Binary Search Tree ADT to include the member function `SingleParentCount` that returns the number of nodes in the tree that have only one child.
32. Write a client function that returns a count of the nodes that contain a value less than the parameter `value`.
33. Extend the Binary Search Tree ADT to include a Boolean function `SimilarTrees` that receives pointers to two binary trees and determines whether the shapes of the trees are the same. (The nodes do not have to contain the same values, but each node must have the same number of children.)
    a. Write the declaration of the function `SimilarTrees` as a `TreeType` member function. Include adequate comments.
    b. Write the body of the function `SimilarTrees`.
34. The `TreeType` member function `MirrorImage` creates and returns a mirror image of the tree.
    a. Write the declaration of the function `MirrorImage`. Include adequate comments.
    b. Write the body of the function `MirrorImage`.
    c. Can the binary tree returned from this function be used for binary searching? If so, how?

35. Write a client function `MakeTree` that creates a binary search tree from the elements in a sorted list of integers. You cannot traverse the list inserting the elements in order, as that would produce a tree that has N levels. You must create a tree with at most $\log_2 N + 1$ levels.
36. Write a client Boolean function `MatchingItems` that determines whether a binary search tree and a sequential list contain the same values.

Examine the following binary search tree and answer the questions in Exercises 37–40. The numbers on the nodes are *labels* so that we can talk about the nodes; they are not key values within the nodes.

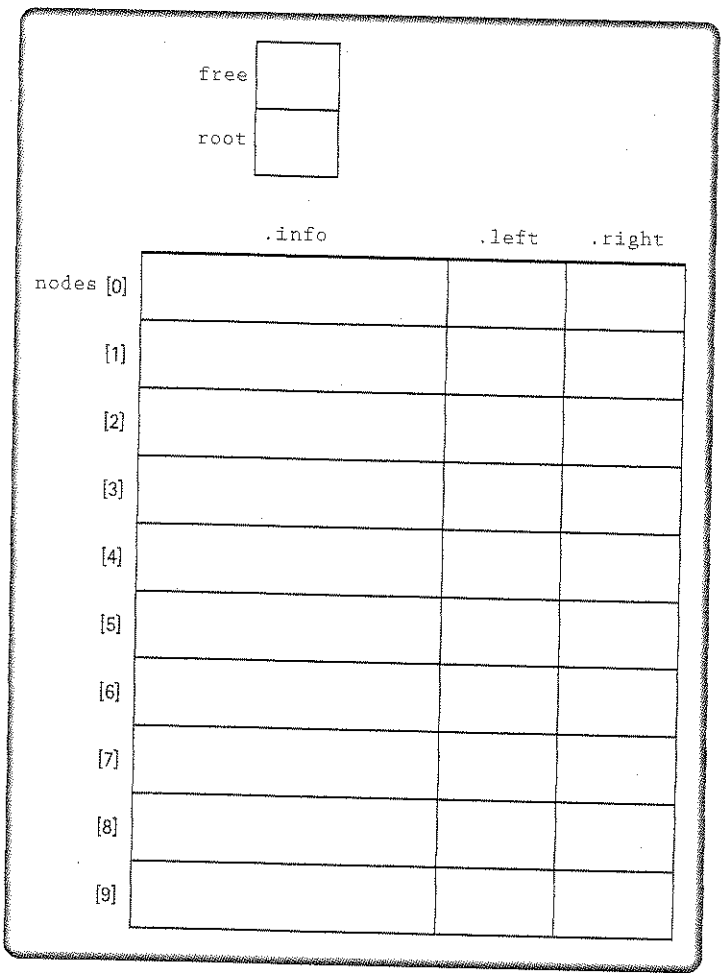


37. If an item is to be inserted whose key value is less than the key value in node 1 but greater than the key value in node 5, where would it be inserted?
38. If node 1 is to be deleted, the value in which node could be used to replace it?
39. 4 2 7 5 1 6 8 3 is a traversal of the tree in which order?
40. 1 2 4 5 7 3 6 8 is a traversal of the tree in which order?
41. In Chapter 6, we discussed how to store a linked list in an array of nodes using index values as "pointers" and managing our list of free nodes. We can use these same techniques to store the nodes of a binary search tree in an array, rather than using dynamic storage allocation. Free space is linked through the `left` member.

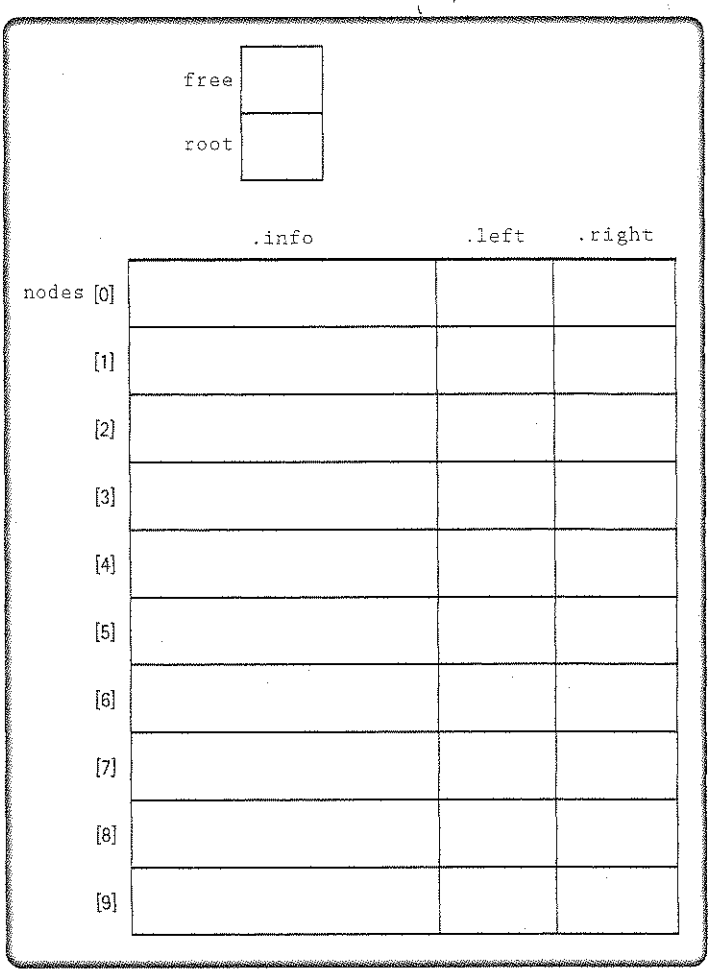
a. Show how the array would look after these elements had been inserted in this order:

Q L W F M R N S

Be sure to fill in all the spaces. If you do not know the contents of a space, use "?".

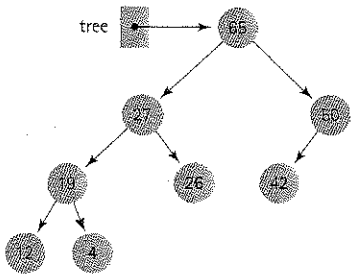


b. Show the contents of the array after "B" has been inserted and "R" has been deleted.

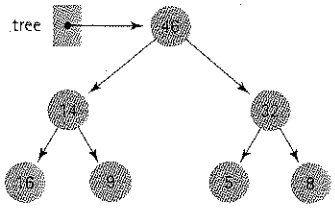


42. a. Which of the following trees are complete?
    b. Which of the following trees are full?

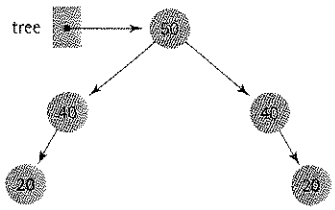
(a)



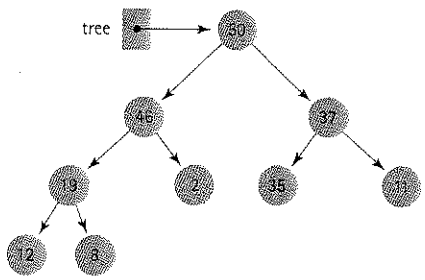
(b)



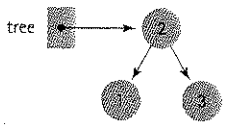
(c)



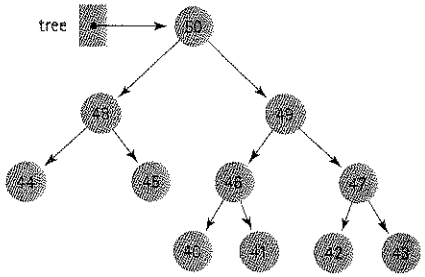
(d)



(e)



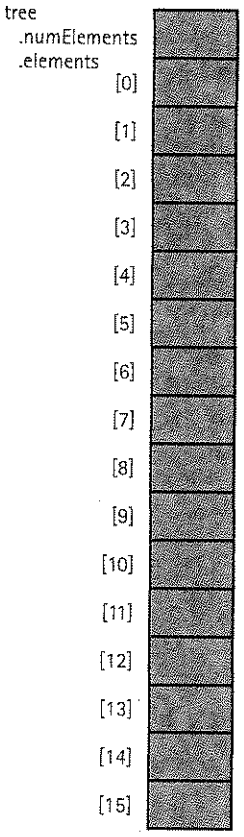
(f)

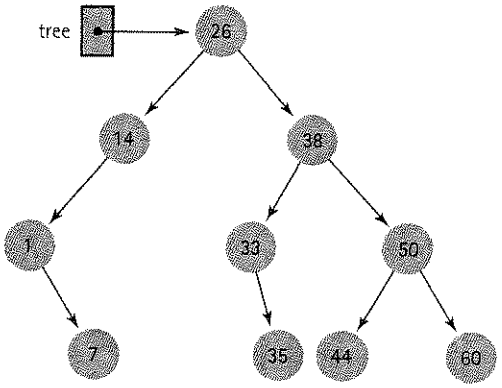


43. The elements in a binary tree are to be stored in an array, as described in the chapter. Each element is a nonnegative `int` value.

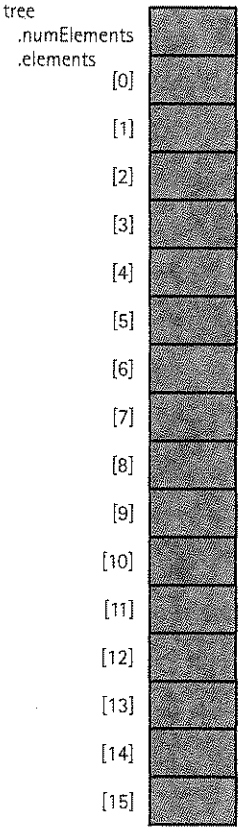
    a. What value can you use as the dummy value, if the binary tree is not complete?

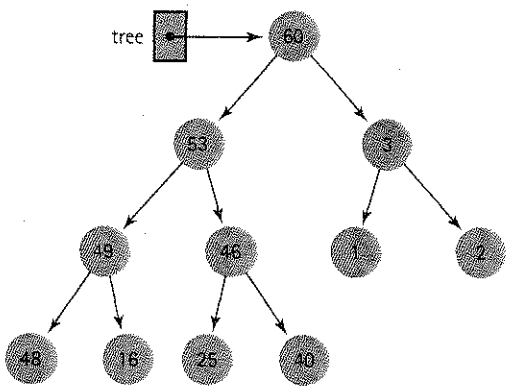
    b. Show the contents of the array, given the following tree.



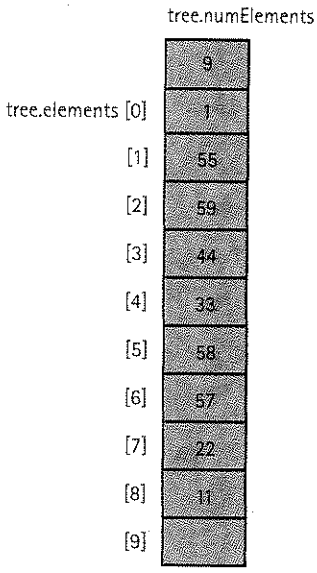


44. The elements in a complete binary tree are to be stored in an array, as described in the chapter. Each element is a nonnegative `int` value. Show the contents of the array, given the following tree.





45. Given the following array, draw the binary tree that can be created from its elements. The elements are arranged in the array as discussed in the chapter.



46. A binary tree is stored in an array called `treeNodes`, which is indexed from 0 to 99, as described in the chapter. The tree contains 85 elements. Mark each of the following statements as True or False, and correct any false statements.
    a. `treeNodes[42]` is a leaf node.
    b. `treeNodes[41]` has only one child.
    c. The right child of `treeNodes[12]` is `treeNodes[25]`.
    d. The subtree rooted at `treeNodes[7]` is a full binary tree with four levels.
    e. The tree has seven levels that are full, and one additional level that contains some elements.
47. Implement the Binary Search Tree ADT as a template class.