CH10

2. Calculate the maximum number of data entries in a:
   a. B-tree of order 5 with a height of 3
   b. B-tree of order 5 with a height of 5
   c. B-tree of order 5 with a height of *h*

4. Draw the B-tree of order 4 created by inserting the following data arriving in sequence:

```
92 24 6 7 11 8 22 4 5 16 19 20 78
```

6. Draw two different B-trees of order 3 that can store seven entries.

8. Create a B+tree of order 5 for the following data arriving in sequence:

```
92 24 6 7 11 8 22 4 5 16 19 20 78
```

10. Using the B-tree of order 3 shown in Figure 10-24, add 50, 78, 101, and 232.

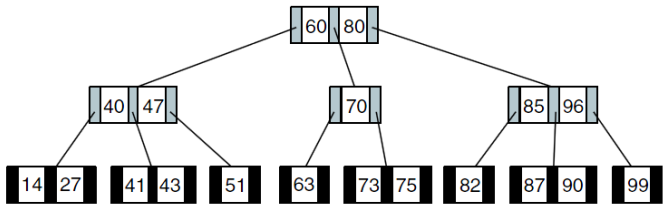

FIGURE 10-24 B-tree for Exercises 10 and 11

CH11

4. Give the depth-first traversal of the graph in Figure 11-23, starting from vertex A.
6. Draw three spanning trees that can be found in the graph in Figure 11-23.
8. Give the adjacency list representation of the graph in Figure 11-23.

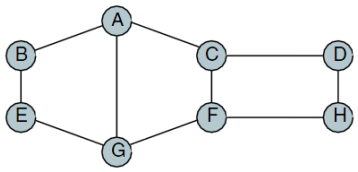

FIGURE 11-23 Graph for Exercises 1 through 8

9. Find the minimum spanning tree of the graph in Figure 11-24.

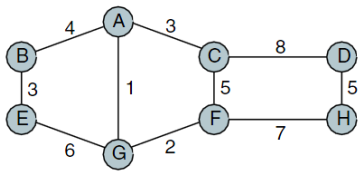

FIGURE 11-24 Graph for Exercises 9 through 12

23. Write the C code for Algorithm 11-13, "Shortest Path," using the ADT given in the text.

ALGORITHM 11-13 Shortest Path

```
Algorithm shortestPath (graph)
Determine shortest path from a network vertex to other
vertices.
    Pre    graph is pointer to network
    Post   minimum path tree determined
```

continued

ALGORITHM 11-13 Shortest Path *(continued)*

```
1 if (empty graph)
  1  return
2 end if
3 loop (through all vertices)
     Initialize inTree flags and path length.
  1  set vertex inTree flag to false
  2  set vertex pathLength to maximum integer
  3  loop (through all edges)
     1  set edge inTree flag to false
     2  get next edge
  4  end loop
  5  get next vertex
4 end loop
  Now derive minimum path tree.
5 set first vertex inTree flag to true
6 set vertex pathLength to 0
7 set treeComplete to false
8 loop (not treeComplete)
  1  set treeComplete to true
  2  set minEdgeLoc to null
  3  set pathLoc    to null
  4  set newPathLen to maximum integer
  5  loop (through all vertices)
        Walk through graph checking vertices in tree.
     1  if (vertex in tree AND outDegree > 0)
        1  set edgeLoc to firstEdge
        2  set minPath to pathLength
        3  set minEdge to maximum integer
        4  loop (through all edges)
              Locate smallest path from this vertex.
           1  if (destination not in tree)
              1  set treeComplete to false
              2  if (edge weight < minEdge)
                 1  set minEdge to edge weight
                 2  set minEdgeLoc to edgeLoc
              3  end if
           2  end if
           3  set edgeLoc to edgeLoc nextEdge
        5  end loop
           Test for shortest path.
        6  if (minPath + minEdge < newPathLen)
           1  set newPathLen to minPath + minEdge
           2  set pathLoc    to minEdgeLoc
        7  end if
     2  end if
     3  get next vertex
  6  end loop
```

continued

ALGORITHM 11-13 Shortest path *(continued)*

```
   7   if (pathLoc not null)
          Found edge to insert into tree.
       1  set pathLoc             inTree flag to true
       2  set pathLoc destination inTree flag to true
       3  set pathLoc destination pathLength to newPathLen
   8   end if
 9 end loop
end shortestPath
```

27. The graph is another structure that can be used to solve the maze problem (see Project 24 in Chapter 3). Every start point, dead end, goal, and decision point can be represented by a node. The arcs between the nodes represent one possible path through the maze. A graph maze is shown in Figure 11-27.

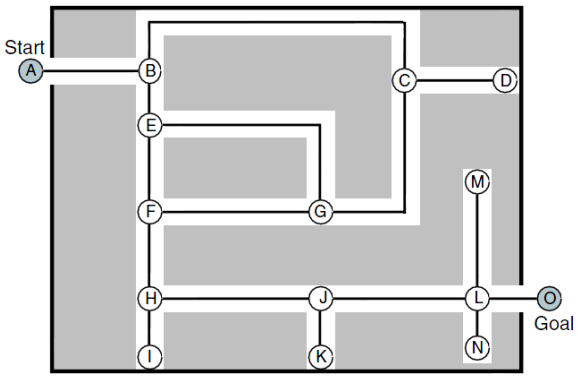

FIGURE 11-27 Graph Maze for Project 27