1. Mark the following statements as true or false.

a. A binary tree must be nonempty. صحيح

b. The level of the root node is 0. صحيح

c. If a tree has only one node, the height of this tree is 0 because the number of levels is 0. صحيح

d. The inorder traversal of a binary tree always outputs the data in ascending order. خاطئ

1. The binary tree of the following Figure is to be used for Exercises 1 through 6.



1. Find LA, the node in the left subtree of A.

L\_(A) = D

1. Find RA, the node in the right subtree of A.

R\_(A) = E

1. Find RB, the node in the right subtree of B.

R\_(B) = G

1. List the nodes of this binary tree in an inorder sequence.

D, A, E, B, G, C, F

1. List the nodes of this binary tree in a preorder sequence.

A, D, B, E, C, G, F

1. List the nodes of this binary tree in a postorder sequence.

E, B, G, F, C, A

1. The binary search tree of the following Figure is to be used for Exercises 1through 4.



1. List the path from the node with info 80 to the node

with info 79.

80, 65, 85,79

1. A node with info 35 is to be inserted in the tree.

List the nodes that are visited by the function

insert to insert 35. Redraw the tree after inserting 35.

1. Delete node 52 and redraw the binary tree.

بعد حذف 25 ، تظل الشجرة دون تغيير حيث أن 25 هي عقدة ورقية

1. Delete node 40 and redraw the binary tree.

80

/ \

65 85

/ \ \

35 75

1. Delete nodes 80 and 58 in that order. Redraw the

binary tree after each deletion.

75

/ \

65 85

/ \ \

35 58

:بعد حذف 80

75

/ \

65 85

/ \

35

1. Write the definition of the function, **nodeCount**, that returns the number of nodes in a binary tree.

public class Node {

int data;

Node left;

Node right;

public Node(int data) {

this.data = data;

left = null;

right = null;

}

}

public class BinaryTree {

Node root;

public BinaryTree() {

root = null;

}

public int nodeCount(Node node) {

if (node == null) {

return 0;

} else {

return 1 + nodeCount(node.left) + nodeCount(node.right);

}

}

}

1. Write the definition of the function, **leavesCount**, that takes as a parameter a reference to the root node of a binary tree and returns the number of leaves in a binary tree.

public class Node {

int data;

Node left;

Node right;

public Node(int data) {

this.data = data;

left = null;

right = null;

}

}

public class BinaryTree {

Node root;

public BinaryTree() {

root = null;

}

public int leavesCount(Node node) {

if (node == null) {

return 0;

} else if (node.left == null && node.right == null) {

return 1;

} else {

return leavesCount(node.left) + leavesCount(node.right);

}

}

}

1. Draw the binary tree representation of the following arithmetic expression:

“(((5+2) ∗ (2−1))/((2+9)+((7−2)−1)) ∗8)”.

\*

/ \

/ \

/ \

/ \

/ \

/ \

/ \

/ \

/ \

/ \

/ \

/ \

/ \

/ \

/ \

+ +

/ \ / \

5 2 + -

/ \

2 1

1. Insert, into an empty binary search tree, entries with keys 30, 40, 24, 58, 48, 26, 11, 13 (in this order). Draw the tree after each insertion.

إدخال القيمة30-1

30

1. إدخال القيمة 40:

30

\

40

1. إدخال القيمة 24:

30

/ \

24 40

1. إدخال القيمة 58:

30

/ \

24 40

\

58

1. إدخال القيمة 48:

30

/ \

24 40

\

58

/

48

1. إدخال القيمة 26:

30

/ \

24 40

\ \

26 58

/

48

1. إدخال القيمة 11:

30

/ \

24 40

\ \

26 58

/ /

11 48

1. إدخال القيمة 13:

30

/ \

24 40

\ \

26 58

/ /

11 48

\

13