Homework

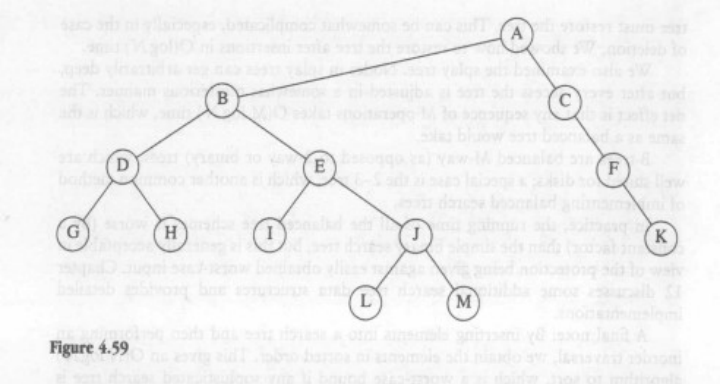


Figure 4.59

Assignment 1 For the tree in Figure 4.59:
a. Which node is the root?

b. Which nodes are leaves?

c. What is the depth of three

Assignment 2 For each node in the tree of Figure 4.59:
a. Name the parent node.
b. List the children.
c. List the siblings.
d. Compute the depth.
e. Compute the height.

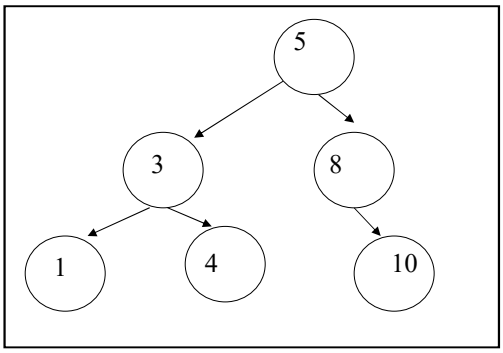
Assignment 3 Please write a C program to create a **binary tree** which contains the number with basic functions:

a. Insert a new node with value X to binary tree as a leaf.
   Notes: Since the given tree is a binary tree, we can insert the element wherever we want. To insert an element, we can use the level order traversal and insert the element wherever we find the node whose left or right child is NULL.
b. Delete a node with value X (You can delete a node in any way you want to satisfy the new tree is binary tree)
c. Print all nodes of a tree with: Pre-order, Post-order, In-order, and BFS.
d. Print the maximum value and minimum value of a binary tree.
e. Write a function for finding the number of full nodes in the binary tree without using recursion.
f. Write a function for finding the height (or depth) of the binary tree without using recursion.
g. Write a function for finding the diameter of the binary tree. The diameter of a tree (sometimes called the *width*) is the number of nodes on the longest path between two leaves in the tree.
h. Write a function for checking the existence of path with given sum. That means, given a sum, check whether there exists a path from root to any of the nodes.

After building all these functions, write a C program to create a binary tree with input data inserted into the tree with this order: 31 45 36 14 52 42 6 21 73 47 26 37 33 8. After that, try to test all your functions.

Assignment 4 Please write a C program to create a **binary search tree (BST)** which contains the number with basic functions:

a. Insert a new node with value X to **BST**. You have to guarantee the new tree is also a **BST**
b. Delete a node with value X. You have to guarantee the new tree is also a **BST**
c. Search a value X in a **BST**
d. Print all nodes of a **BST** with: Pre-order, Post-order, In-order, and BFS.
e. Print all nodes of a **BST** with increasing order
f. Print the maximum value and minimum value of a BST
g. Given two values, determine these two values already exist in in **BST**, if yes, find the lowest common ancestor (*LCA*). For example, given the following BST, the LCA of 1 and 8 is 5, the LCA of 1 and 4 is 3.



After building all these functions, write a C program to create a **BST** with input data inserted into the BST with this order: 31 45 36 14 52 42 6 21 73 47 26 37 33 8. After that, try to test all your functions.