**King Fahd University of Petroleum & Minerals**

**College of Computer Science and Engineering**

**Information and Computer Science Department**

**ICS 202 – Data Structures**

# Lab 06: Binary Trees

**Objectives**

The objective of this lab is to design, implement and use binary trees.

**Outcomes**

After completing this Lab, students are expected to:

• Understand classes for binary trees.

• Implement methods for binary trees.

• Traverse binary trees (breadth-first, depth-first)

• Insert into and delete from binary trees.

**BinaryTree insertion and deletion**

There are no fixed rules for inserting and deleting from a Binary-tree. In our Binary-tree implementation, we use the insertion and deletion algorithms given below:

* Given a binary tree and a key, insert the key into the binary tree at the first position available in level order traversal.



Note: We can create a Binary tree without using the insert method. We do this by creating the root node and then linking it with other nodes:

 BinaryTree<Integer> tree = new BinaryTree<Integer>();

 tree.root= new BTNode(7);

 tree.root.left= new BTNode(40);

 tree.root.left.right = new BTNode(28);

 

* Given a binary tree, delete a node from it by making sure that tree shrinks from the bottom (i.e. the deleted node is replaced by the last leaf node).

 ****

**Level of a Binary tree Node**

The Level (or depth) of a node v: The length of the path from the root to v (i.e., the number of edges from the root to v).

 

**Note:**

**For the purpose of this lab you are allowed to use the given BinaryTree, BTNode, and BinaryTreeDriver classes only.**

**Lab Exercises**

1. Write a method **public int count()** to count the number of nodes in a binary tree.
2. Write a method **public boolean isLeaf(BTNode node)** to determine if a given binary tree node is a leaf.

1. Write a method **public int countLeaves()** to count the number of leaves in a binary tree.
2. Write a method **int getLevel(T data)** to find the level of a node with key **data** of a binary tree. Assume that the binary tree has distinct keys.

Test program.

Write a test program that creates the binary tree shown below, traverses it using the breadth-first and depth-first traversals (preorder, inorder, and postorder) and prints the traversal results. It also tests the **delete** method and the above methods. For example, for the following tree:

 

a sample program run is:

 The number of nodes in the tree is 6
MMThe number of leaf nodes in the tree is 3
MMThe level of node with key 4 is 2
MMTrying to find the level of node with key 60 ...
MMjava.util.NoSuchElementException: Key not in tree.
MM

 Preorder Traversal is:
MM1 2 4 5 12 3

MMInorder Traversal is:
MM4 2 12 5 1 3

MMBefore deleting key 3, level order traversal of binary tree is:
MM1 2 3 4 5 12
MMThe tree is:
MM R----1
MMM L----2
MM M | L----4
MM M | R----5
MM M | L----12
MM M R----3
MM
MMAfter deleting key 3, level order traversal of binary tree is:
MM1 2 12 4 5
MMThe tree is:
MM MR----1
MM M L----2
MM M | L----4
MM M | R----5
MM M R----12
MMM