Mohammad Ali Jinnah University

Chartered by Government of Sindh - Recognized by HEC

Lab Task 9

Name: Muhamad Fahad

Id: FA19-BSSE-0014

Subject: Data Structures and Algorithms Lab (CS 2511)

Lab Title: Trees

Section: AM

Teacher: MUHAMMAD MUBASHIR KHAN

Date: Monday, January 4, 2021

1. **Implement Inorder, Preorder and Postorder techniques.**
2. **Create an insertion method for a complete binary tree.**

Code:

```
package com.company.Tree;

public class Starting {
    public static void main(String[] args) {
        BinaryTree bt = new BinaryTree();

        int arr[] = {1, 2, 3, 4, 5, 7,5,3,5,77,1,3,4,12,5,12};
        bt.root = bt.insertLevelOrder(arr, bt.root, 0);

        System.out.print("\nInorder: ");
        bt.Inorder(bt.root);

        System.out.print("\nPostorder: ");
        bt.Postorder(bt.root);

        System.out.print("\nPreorder: ");
        bt.Preorder(bt.root);
    }
}

class BinaryTree{
    public static Node root;

    static class Node{
        int value;
        Node left, right;

        public Node(int item) {
            value = item;
            left = right = null;
        }
    }

    void Postorder(Node node) {
        if (node == null)
            return;

        Postorder(node.left);
        Postorder(node.right);

        System.out.print(node.value + " ");
    }
    void Inorder(Node node) {
        if (node == null)
            return;

        Inorder(node.left);
        System.out.print(node.value + " ");
        Inorder(node.right);
    }
```

```
    void Preorder(Node node) {
        if (node == null)
            return;

        System.out.print(node.value + " ");
        Preorder(node.left);
        Preorder(node.right);
    }

    BinaryTree() {
        root = null;
    }

    public Node insertLevelOrder(int[] arr, Node root, int Count) {
        if (Count < arr.length) {
            Node temp = new Node(arr[Count]);
            root = temp;

            Count *= 2;
            root.left = insertLevelOrder(arr, root.left, ++Count);
            root.right = insertLevelOrder(arr, root.right, ++Count);
        }
        return root;
    }
}
```

Output:

```
Inorder: 12 3 4 5 2 77 5 1 1 3 7 4 3 12 5 5
Postorder: 12 3 5 4 77 1 5 2 3 4 7 12 5 5 3 1
Preorder: 1 2 4 3 12 5 5 77 1 3 7 3 4 5 12 5
```