Binary Tree Questions

1.Post Order Iterative Solution to be Build

CODE:

```
package com.selflearning.Prac;
import java.util.Stack;

class Node
{
    int data;
    Node left, right;

    Node(int item)
    {
        data = item;
        left = right;
    }
}
public class PostOrder
{
    Node root;
    private void postOrderIterative(Node root) {
        Stack<Node> stack = new Stack<>();
        while(true) {
            while(root != null) {
                stack.push(root);
                stack.push(root);
                root = root.left;
            }
            if(stack.empty()) return;
            root = stack.pop();
            if(!stack.empty() && stack.peek() == root) root =
root.right;
              else {
                      System.out.print(root.data + " "); root =
null;
            }
        }
    }
```

2.Print Nodes at Distance K from the Root

CODE:

class Solution

{

```
  int printKDistantfromLeaf(Node root, int k)
  {
     ArrayList<Node> path = new ArrayList<>();
     HashSet<Node> uniqueNodes = new HashSet<>();
     findPath(root,k,path,uniqueNodes);

     return uniqueNodes.size();

  }

  static void findPath(Node root,int K,ArrayList<Node> path,HashSet<Node> uniqueNodes){
     if(root == null)
        return;

     path.add(root);
     if(root.left == null && root.right == null){
        if(K<path.size())
           uniqueNodes.add(path.get(path.size()-K-1));
     }

     findPath(root.left,K,path,uniqueNodes);
     findPath(root.right,K,path,uniqueNodes);
     path.remove(root);
  }
}
```

3. Maximum Element in a Binary Tree.

CODE:

```
class Node {
    int data;
    Node left, right;
```

```
    public Node(int data)
    {
        this.data = data;
        left = right = null;
    }
}

class BinaryTree {
    Node root;


    static int findMax(Node node)
    {
        if (node == null)
            return Integer.MIN_VALUE;

        int res = node.data;
        int lres = findMax(node.left);
        int rres = findMax(node.right);

        if (lres > res)
            res = lres;
        if (rres > res)
            res = rres;
        return res;
    }
}
```