Assignment No.2

Roll No:2460

```
import java.util.*;

class Node{
    int data;
    Node left;
    Node right;
    Node(int d){
        this.data=d;
        left=null;
        right=null;
    }
}



public class Ass234 {

    // Insert the node in BST
  public static Node Insert(Node root,int data){
      Node newn=new Node(data);
      if(root==null){
          root=newn;
      }
      else{
          if(data<root.data)
          root.left=Insert(root.left,data);
          else
          root.right=Insert(root.right,data);

        }
        return root;
  }



// Inorder traversal of the BST
  public static void Inorder(Node root){
 if(root!=null){
     Inorder(root.left);
     System.out.print(root.data+" ");
     Inorder(root.right);
 }
```

```
// get the height of the BST
  public static int height(Node root){
    int h=0;
    if(root==null)
    return 0;
    else{
      int l=height(root.left);
      int r=height(root.right);
      if(l>r)
      h=l+1;
      else
      h=r+1;
    }
    return h;
  }

//Level order traversal of BST
  public static void level(Node root){
Queue<Node> q=new LinkedList<Node>();
while(root!=null){
  System.out.print(root.data+" ");
  if(root.left!=null)
  q.add(root.left);
  if(root.right!=null)
  q.add(root.right);
  if(!q.isEmpty()){
    root=q.poll();
 }
  else{
    root=null;
  }
 }
}

//count the total number of leaf nodes in BST
  public static int count(Node root){
    int c=0;
    if(root==null)
    return 0;
    else if(root.left==null&&root.right==null){
      return 1;
    }

     c+=count(root.left);
      c+=count(root.right);
    return c;
  }
```

```java
//print BST in descending order
 public static void desc(Node root){
    if(root==null)
    return;
    desc(root.right);
    System.out.print(root.data+" ");
    desc(root.left);
  }

//get the parent node of the given node
  public static int parent(Node root,int d){

    Queue<Node> q=new LinkedList<Node>();
       while(root!=null){
        if(root.left!=null&&root.left.data==d)
          return root.data;

         else if(root.right!=null &&root.right.data==d)
           return root.data;

         else{
           if(root.left!=null&&root.data>d)
           q.add(root.left);
           else if(root.right!=null&&root.data<d)
           q.add(root.right);
           if(!q.isEmpty()){
             root=q.poll();
           }
           else
           root=null;
         }
       }
       return 0;
  }

//get the minimum node in BST
  public static Node min(Node root){
    if(root.left==null)
    return root;
return   min(root.left);
  }

//get the maximum node in BST
  public static Node max(Node root){
    if(root.right==null)
    return root;
    return max(root.right);
  }
```

```java
public static void main(String[] args) {

        Node root=null;

       int t=0;

       Scanner sc=new Scanner(System.in);

          System.out.println("Enter total number of nodes");
          t=sc.nextInt();

          while(t-->0){
            System.out.println("Enter data");
              int val=sc.nextInt();
              root=Insert(root,val);
            }

            System.out.print("Inorder Traversal : ");
            Inorder(root);

            System.out.println("\nMinimum node : "+min(root).data);

            System.out.println("Maximum node : "+max(root).data);

            System.out.print("Level Order Traversal : ");
            level(root);

            System.out.println("\nHeight of tree : "+height(root));

            System.out.print("Tree in descending order : ");
            desc(root);

            System.out.print("\nEnter a node for finding it's parent : ");
            int d=sc.nextInt();


            System.out.println("So the parent of node "+d
            +" is "+ parent(root,d));

            System.out.print("\nNumber Of Leaf Nodes : "+count(root));

            sc.close();
    }
}
```