Name- VARANASI. KASYAP

DSA ASSIGNMENT

Question 1-

Write a program to insert elements in to an empty BST and perform following operations.

i) static int max() - returns greatest element from BST

ii) static int min()- returns smallest element from BST

iii) static int count_leaf()- returns number of leaves in BST

Code-

```
public class BSTminmaxcount {
   public static class TreeNode
   {
      int data;
      TreeNode left;
      TreeNode right;
      TreeNode(int data)
      {
         this.data=data;
      }
   }

   public static boolean search(TreeNode root,TreeNode nodeToBeSearched)
   {
      if(root==null)
         return false;
      if(root.data== nodeToBeSearched.data)
      {
         return true;
```

```
    }
    boolean result=false;
    if(root.data > nodeToBeSearched.data)
        result=search(root.left,nodeToBeSearched);
    else if(root.data < nodeToBeSearched.data)
        result= search(root.right,nodeToBeSearched);
    return result;
}
    public static  int getLeafCountOfBinaryTree(TreeNode node)
{
    if(node == null)
        return 0;
    if(node.left ==null && node.right==null)
        return 1;
    else
        return getLeafCountOfBinaryTree(node.left)+ getLeafCountOfBinaryTree(node.right);
}
public static TreeNode minimumElement(TreeNode root)
{
    if(root.left==null)
        return root;
    else
    {
        return minimumElement(root.left);
    }
}

public static TreeNode maximumElement(TreeNode root)
{
    if(root.right==null)
        return root;
```

```
    else
    {
        return maximumElement(root.right);
    }
}
public static TreeNode insert(TreeNode root,TreeNode nodeToBeInserted)
{
    if(root==null)
    {
        root=nodeToBeInserted;
        return root;
    }

    if(root.data > nodeToBeInserted.data)
    {
        if(root.left==null)
            root.left=nodeToBeInserted;
        else
            insert(root.left,nodeToBeInserted);
    }
    else if(root.data < nodeToBeInserted.data)
        if(root.right==null)
            root.right=nodeToBeInserted;
        else
            insert(root.right,nodeToBeInserted);
    return root;
}

public static void inOrder(TreeNode root)
{
    if(root==null)
```

```
        return;
    inOrder(root.left);
    System.out.print(root.data+" ");
    inOrder(root.right);
  }
  public static void main(String[] args)
  {



    TreeNode rootNode=createBinarySearchTree();
    System.out.println("Minimum element in binary search tree:
"+minimumElement(rootNode).data);
    System.out.println("Maximum element in binary search tree:
"+maximumElement(rootNode).data);
    System.out.println("Number of leaf nodes in binary tree
:"+getLeafCountOfBinaryTree(rootNode));
  }


  public static TreeNode createBinarySearchTree()
  {
    TreeNode rootNode =new TreeNode(40);
    TreeNode node20=new TreeNode(20);
    TreeNode node10=new TreeNode(10);
    TreeNode node30=new TreeNode(30);
    TreeNode node60=new TreeNode(60);
    TreeNode node50=new TreeNode(50);
    TreeNode node70=new TreeNode(70);



    insert(null,rootNode);
    insert(rootNode,node20);
    insert(rootNode,node10);
```

```
        insert(rootNode,node30);
        insert(rootNode,node60);
        insert(rootNode,node50);
        insert(rootNode,node70);

        rootNode.left=node20;
        rootNode.right=node60;

        node20.left=node10;
        node20.right=node30;

        node60.left=node50;
        node60.right=node70;
        return rootNode;
    }

}
```

Output-