Name – Yash Daga Roll Number – 20BCE7323

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

Note: you can use any of the tree traversal technique to display items.

Code:

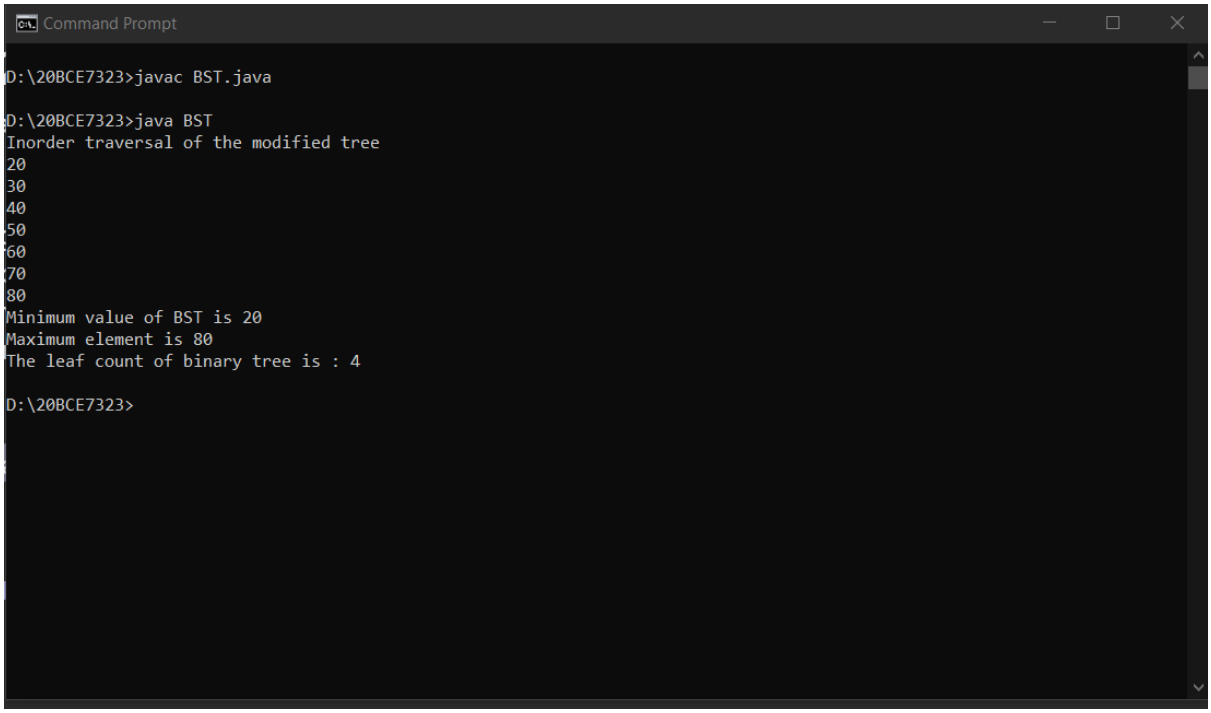
```
class Node
{
   int data;
   Node left, right;

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

public class BST
{
        Node root;
   BST()
   {
       root = null;
```

```
}
int count_leaf()
{
   return count_leaf(root);
}
      void insert(int key)
{
    root = insertRec(root, key);
}
      Node insertRec(Node root, int data)
{

   if (root == null)
   {
      root = new Node(data);
      return root;
   }
    if (data < root.data)
      root.left = insertRec(root.left, data);
   else if (data > root.data)
      root.right = insertRec(root.right, data);

   return root;
}

int count_leaf(Node node)
```

```
{
    if (node == null)
        return 0;
    if (node.left == null && node.right == null)
        return 1;
    else
        return count_leaf(node.left) + count_leaf(node.right);
}
      int min(Node node) {
    Node current = node;

    /* loop down to find the leftmost leaf */
    while (current.left != null) {
        current = current.left;
    }
    return (current.data);
}
      int max(Node node)
{
    if (node == null)
        return Integer.MIN_VALUE;

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

```
    if (lres > res)
        res = lres;
    if (rres > res)
        res = rres;
    return res;
}
      void inorder()
{
    inorderRec(root);
}
      void inorderRec(Node root)
{
    if (root != null) {
        inorderRec(root.left);
        System.out.println(root.data);
        inorderRec(root.right);
    }
}
public static void main(String args[])
{
                BST tree = new BST();
    tree.insert(50);
    tree.insert(30);
    tree.insert(20);
    tree.insert(40);
    tree.insert(70);
```

```
        tree.insert(60);
        tree.insert(80);
                System.out.println(
           "Inorder traversal of the modified tree");
        tree.inorder();
        System.out.println("Minimum value of BST is " + tree.min(tree.root));
                System.out.println("Maximum element is "+ tree.max(tree.root));
        System.out.println("The leaf count of binary tree is : " + tree.count_leaf());

    }
}
```




```
80      }
81      void inorderRec(Node root)
82      {
83          if (root != null) {
84              inorderRec(root.left);
85              System.out.println(root.data);
86              inorderRec(root.right);
87          }
88      }
89      public static void main(String args[])
90      {
91          BST tree = new BST();
92          tree.insert(50);
93          tree.insert(30);
94          tree.insert(20);
95          tree.insert(40);
96          tree.insert(70);
97          tree.insert(60);
98          tree.insert(80);
99          System.out.println(
100             "Inorder traversal of the modifi
101         tree.inorder();
102         System.out.println("Minimum value of
103         System.out.println("Maximum element is "+ tree.max(tree.root));
104         System.out.println("The leaf count of binary tree is : " + tree.count_leaf());
105
106     }
107 }
108
```