LAB SESSION - 7

Pre-lab

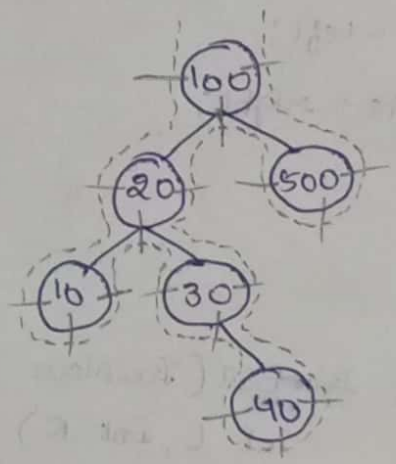
1.

```
package lab7;
public class BinarySearchTree{
    class Node{
        int key;
        Node left, right;
        public Node(int item)
        {
            key = item;
            left = right = null;
        }
    }
    Node root;
    BinarySearchTree(){
        root = null;
    }
    BinarySearchTree(int value){
        root = new Node(value);
    }
    void insert(int key){
        root = insertRec(root, key);
    }
    Node insertRec(Node root, int key){
        if(root == null){
            root = new Node(key);
```

```
        return root;
    }
    else if (key < root.key)
        root.left = insertRec(root.left, key);
    else if (key > root.key)
        root.right = insertRec(root.right, key);
    return root;
}
void inOrder() {
    inOrderRec(root);
}
void inorderRec(Node root) {
    if (root != null) {
        inorderRec(root.left);
        System.out.println(root.key);
        inOrderRec(root.right);
    }
}
public static void main(String args[]){
    BinarySearchTree tree = new
                BinarySearchTree();
    tree.insert(50);
    tree.insert(30);
    tree.insert(20);
    tree.insert(40);
    tree.insert(70);
    tree.insert(60);
```

```
        tree.insert(80);
        tree.inorder();
    }
}
```

2. Given tree,



Preorder traversal:

100, 20, 10, 30, 40, 500

Inorder traversal:

10, 20, 30, 40, 100, 500

Postorder traversal:

10, 40, 30, 20, 500, 100

level order traversal:

100, 20, 500, 10, 30, 40

In-Lab:

1.
```
public class TreeNode{
    int val;
    TreeNode left, right;
    TreeNode(){
    }
```

```
    TreeNode(int val){
        this.val = val;
    }
    TreeNode (int val, TreeNode left, TreeNode right)
    {
        this.val=val;
        this.left = left;
        this.right = right;
    }
}
class Solution {
    public TreeNode trimBST(TreeNode root,
                            int L, int R)
    {
        if (root == null)
            return null;
        else if(root.val >=L && root.val <= R){
            root.left = trimBST(root.left, L, R);
            root.right = trimBST(root.right, L, R);
        }
        else if(root.val<L)
            root = trimBST(root.right, L,R);
        else if (root.val >R)
            root = trimBST(root.left, L,R);
        return root;
    }
}
```

Sample Input: [3,0,4, null, 2, null, null, 1], low = 1
high = 3

Sample output: [3, 2, null, 1]

2.
```
package Lab7;
public class Node {
    int data;
    Node left, right;
    Node(int x) {
        data = x;
        left = right = null;
    }
}
```

```
package Lab7;
import java.io.*;
public class SmallestEle{
    static int count = 0;
    public static Node insert(Node root, int x)
    {
        if (root == null)
            return new Node(x);
        if (x < root.data)
            root.left = insert(root.left, x);
        else if (x > root.data)
            root.right = insert(root.right, x);
        return root;
    }
```

```
public static Node KthSmallest(Node root, int k)
{
    if(root == null)
        return null;
    Node left = KthSmallest(root.left, k);
    if(left != null)
        return left;
    count++;
    if(count == k)
        return root;
    return KthSmallest(root.right, k);
}
public static void main(String args[])
{
    Node root = null;
    int keys[] = {20, 8, 22, 4, 12, 10, 14};
    for(int x: keys)
        root = insert(root, x);
    int k = 3;
    Node res = KthSmallest(root, k);
    if(res == null)
        System.out.println("There are less
            than k nodes in BST");
    else
        System.out.println("K'th Smallest
            element is: " + res.data);
}
}
```

Output: K-th Smallest element is : 10

3.

```
public class TreeNode {
    int val;
    TreeNode left, right;
    TreeNode(int x) {
        val = x;
    }
}
public class Solution {
    public boolean hasPathSum(TreeNode root,
                              int sum) {
        if(root == null)
            return false;
        if (root.left == null && root.right == null)
            return sum = root.val;
        return hasPathSum(root.left, sum-root.val)
            || hasPathSum(root.right, sum - root.val);
    }
}
```

Sample Input:

[3,9,20, null, null, 15, 7]

Sample Output:

3

Post lab

1.

```
package Lab7;
public class InorderSuccessorPredecessor {
    static int successor, predecessor;
    public void successorPredecessor(Node root,
                                     int val)
    {
        if (root != null) {
            if (root.data == val) {
                if (root.left != null) {
                    Node t = root.left;
                    while (t.right != null)
                        t = t.right;
                    predecessor = t.data;
                }
                if (root.right != null) {
                    Node t = root.right;
                    while (t.left != null)
                        t = t.left;
                    successor = t.data;
                }
            }
            else if (root.data > val) {
                successor = root.data;
                successorPredecessor(root.left, val);
            }
```

```
            else if (root.data<val){
                Predecessor = root.data;
                successorPredecessor(root.right,val);
            }
        }
    }
    public static void main(String args[]){
        Node root = new Node(20);
        root.left = new Node(10);
        root.right = new Node(30);
        root.left.left = new Node(5);
        root.left.left.right = new Node(7);
        root.left.right = new Node(15);
        root.right.left = new Node(25);
        root.right.right = new Node(35);
        root.left.right.left = new Node(13);
        root.left.right.right = new Node(18);
        InorderSuccessorPredecessor i = new
                    InorderSuccessorPredecessor();
        i.successorPredecessor(root,10);
        System.out.println("Inorder successor of
            10 is: "+ successor +" and predecessor
            is "+ predecessor);
    }
}
public class Node {
    int data;
```

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

Output:

Inorder Successor of 10 is : 13 and predecessor is : 7

2.

```
public class TreeNode{
    int val;
    TreeNode left, right;
    TreeNode()
    {
    }
    TreeNode(int val){
        this.val=val;
    }
    TreeNode (int val, TreeNode left, TreeNode
                                        right)
    {
        this.val=val;
        this.left= left;
        this.right= right;
```

```
    }
}
public class Solution{
    private int sum=0;
    public TreeNode convertBST(TreeNode root)
    {
        if (root != null){
            convertBST (root.right);
            sum += root.val;
            root.val = sum;
            convertBST(root.left);
        }
        return root;
    }
}
```

Sample Input:

[4,1,6,0,2,5,7,null,null,null,3,null,null,null,8]

Output:

[30,36,21,36,35,26,15,null,null,null,33,null,
null,null,8].