2100032489

Lab-7

Pre-lab:

1A)
```
Package p1;
Public Node Search(Node root, int key){
    if(root == null || root.key == key)
        return root;
    if(root.key < key)
        return search(root.right, key);
    return search(root.left, key);
}
```

2A)
```
Package p1;
class Node{
    int key;
    Node left, right;
    Public Node(int item){
        key = item;
        left = right = null;
}   }
class BinaryTree{
    Node root;
    BinaryTree(){
        root = null;
    }
void printPostOrder(Node node){
```

```
if (node == null)
    return;
printPostOrder(node.left);
printPostOrder(node.right);
System.out.print(node.key + " ");
}
void printInorder(Node node){
    if (node == null)
        return;
    printInorder(node.left);
    System.out.println(node.key + " ");
    printInorder(node.right);
}
void printPreorder(Node node){
    if (node == null)
        return;
    System.out.println(node.key + " ");
    printPreorder(node.left);
    printPreorder(node.right);
}
public static void main(String[] args){
    BinaryTree tree = new BinaryTree();
    tree.root = new Node(1);
    tree.root.left = new Node(2);
    tree.root.right = new Node(3);
    System.out.println("preorder traversal");
    tree.printPreorder();
}}
```

21000328489.

In-lab:

1A) class Solution{

Public TreeNode trimBST(TreeNode R, int L, int H){

if(R == null) return R;

if(R.val < L)

return trimBST(R.right, L, H);

else if (R.val > H) return trimBST(R.left, L, H);

R.left = trimBST(R.left, L, H);

R.right = trimBST(R.right, L, H);

}} return R;

2A) import java.io.*;

class Node{

int data;

Node left, right;

Node(int x){

data = x;

left = right = null;

}}

class GFG{

static int count = 0;

Public static Node insert(Node root, int x){

if(root == null)

```
    return new Node(x);
}
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++;
}
public static void main(String[] args){
    Node root = null;
    int keys[] = {20, 30, 8, 22, 4, 12, 14};
    for(int x: keys)
        root = insert(root, x);
    int k = 3;
    printKthSmallest(root, k);
}}
```

3A)

```
package p1;
class Node{
    int data;
    Node left, right;
    Node(int item){
        data = item;
```

```
2100032489
    left = right = null;
}}
class BinaryTree{
    Node root;
boolean hasPathSum(Node node, int sum){
    if(root == null)
        return false;
    boolean ans = false;
    int subSum = sum - node.data;
    if(subSum == 0 && node.left == null
        && node.right == null)
        return (ans = true);
    if (node.left != null)
        ans = ans || hasPathSum(node.left, subSum);
}
Public static void main(String [] args){
    int sum = 21;
    BinaryTree tree = new BinaryTree();
    tree.root = new node (10);
    tree.root.left = new Node (8);
    tree.root.right = new Node (3);
    if ( tree.hasPathSum(tree.root, sum))
        System.out.println("leaf path" + sum);
}}
```

Post-lab:

1-A)

```
Package P1;
class CrFCr{
  static class Node{
    int key;
    Node left, right;
    public Node(){}
    public Node(int key){
      this.key = key;
  }; }
static void findPreSuc(Node root, int key){
  if(root == null)
    return;
  if(root.key == key){
  if(root.left != NULL){
    Node tmp = root.left;
    tmp = tmp.right;
  } pre = tmp;
if(root.key > key){
  suc = root;
  findPreSuc(root.left, key);
}
else{
  pre = root;
  findPreSuc(root.right, key);
}}
```

2100032489

```
public static void main(String[] args){
    int key = 65;
    Node root = new Node();
    root = insert(root, 50);
    insert(root, 30);
    insert(root, 20);
    insert(root, 40);
    insert(root, 50);
    findPreSuc(root, key);
}}
```

24)

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