Java Solution:

Set A:

class BTNode {

int elem;

BTNode left, right;

public BTNode(int elem) {

this.elem = elem;

this.left = null;

this.right = null;

}

}

public class BinaryTree {

public static int find(BTNode root, int value) {

if (root == null) {

return 0;

} else {

if (root.elem == value) {

return 0;

} else if (root.elem > value) {

return root.elem + find(root.left, value);

} else if (root.elem < value) {

return root.elem + find(root.right, value);

}

}

return 0;

}

public static void main(String[] args) {

BTNode root = new BTNode(30);

BTNode n1 = new BTNode(10);

BTNode n2 = new BTNode(40);

root.left = n1;

root.right = n2;

BTNode n3 = new BTNode(3);

BTNode n4 = new BTNode(15);

n1.left = n3;

n1.right = n4;

BTNode n5 = new BTNode(35);

BTNode n6 = new BTNode(55);

n2.left = n5;

n2.right = n6;

BTNode n7 = new BTNode(2);

n3.left = n7;

BTNode n8 = new BTNode(36);

n5.right = n8;

int result = find(root, 15);

System.out.println("Result: " + result);

}

}

Set B:

class BTNode {

int elem;

BTNode left, right;

public BTNode(int elem) {

this.elem = elem;

this.left = null;

this.right = null;

}

}

public class BinaryTree {

public static void route(BTNode root, int dest) {

if (root == null) {

return;

} else {

if (root.elem == dest) {

return;

} else if (root.elem > dest && root.left != null) {

System.out.println("Go left");

route(root.left, dest);

} else if (root.elem < dest && root.right != null) {

System.out.println("Go right");

route(root.right, dest);

} else {

System.out.println(dest + " does not exist");

}

}

}

public static void main(String[] args) {

BTNode root = new BTNode(30);

BTNode n1 = new BTNode(10);

BTNode n2 = new BTNode(40);

root.left = n1;

root.right = n2;

BTNode n3 = new BTNode(3);

BTNode n4 = new BTNode(15);

n1.left = n3;

n1.right = n4;

BTNode n5 = new BTNode(35);

BTNode n6 = new BTNode(55);

n2.left = n5;

n2.right = n6;

BTNode n7 = new BTNode(2);

n3.left = n7;

BTNode n8 = new BTNode(36);

n5.right = n8;

route(root, 36);

route(root, 60);

}

}

Rubric:

2.5 Marks - Construct the Node class

2.5 Marks - Construct the BST

1 Marks - defining the function with correct parameters

1.5 Marks - Right base condition

3 Marks - Correct Recursive calls

3 Marks - Correct Calculation (summation for set A and correct conditions for set B )

1.5 Marks - Correct Output statements