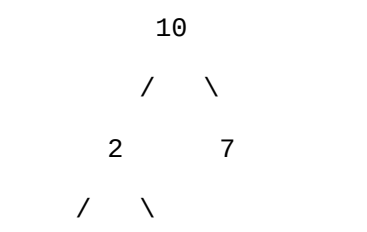
Assignment Questions 21

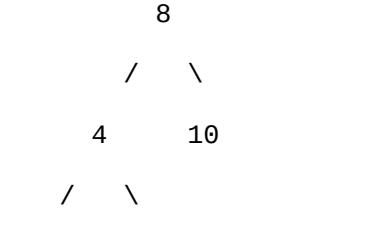
Question-1

You are given a binary tree. The binary tree is represented using the TreeNode class. Each TreeNode has an integer value and left and right children, represented using the TreeNode class itself. Convert this binary tree into a binary search tree.

Input:



Output:



code:-

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

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

class BinaryTree {
      Node root;
      int isSumProperty(Node node)
      {
            int left_data = 0, right_data = 0;
            if (node == null
                  || (node.left == null && node.right == null))
                  return 1;
            else {
                  if (node.left != null)
                        left_data = node.left.data;
                  if (node.right != null)
                        right_data = node.right.data;
                  if ((node.data == left_data + right_data)
                        && (isSumProperty(node.left) != 0)
                        && isSumProperty(node.right) != 0)
```

```
                        return 1;
                  else
                        return 0;
            }
      }

      public static void main(String[] args)
      {
            BinaryTree tree = new BinaryTree();
            tree.root = new Node(10);
            tree.root.left = new Node(8);
            tree.root.right = new Node(2);
            tree.root.left.left = new Node(3);
            tree.root.left.right = new Node(5);
            tree.root.right.right = new Node(2);
            if (tree.isSumProperty(tree.root) != 0)
                  System.out.println("The given tree satisfies children"+ " sum
property");
            else
                  System.out.println("The given tree does not satisfy children"+ "
sum property");
      }
}
```

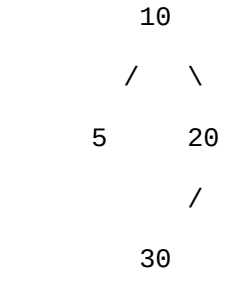
Question-2:
Given a Binary Search Tree with all unique values and two keys. Find the distance between two nodes in BST. The given keys always exist in BST.
Example:
Input-1:
n = 9
values = [8, 3, 1, 6, 4, 7, 10, 14,13]
node-1 = 6
node-2 = 14
Output-1:
The distance between the two keys = 4
Input-2:
n = 9
values = [8, 3, 1, 6, 4, 7, 10, 14,13]
node-1 = 3
node-2 = 4
Output-2:
The distance between the two keys = 2

code:-

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

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

class BinarySearchTree {
      Node root;

      // Constructor
```

```
	BinarySearchTree() {
		root = null;
	}
	Node insert(Node node, int key) {
		// If the tree is empty, return a new node
		if (node == null) {
			node = new Node(key);
			return node;
		}
		if (key < node.key)
			node.left = insert(node.left, key);
		else if (key > node.key)
			node.right = insert(node.right, key);
		return node;
	}
	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);
	}
	public static void main(String[] args) {
		BinarySearchTree tree = new BinarySearchTree();
		tree.root = tree.insert(tree.root, 50);
		tree.insert(tree.root, 30);
		tree.insert(tree.root, 20);
		tree.insert(tree.root, 40);
		tree.insert(tree.root, 70);
		tree.insert(tree.root, 60);
		tree.insert(tree.root, 80);
		int key = 6;
		if (tree.search(tree.root, key) == null)
			System.out.println(key + " not found");
		else
			System.out.println(key + " found");

		key = 60;
		if (tree.search(tree.root, key) == null)
			System.out.println(key + " not found");
		else
			System.out.println(key + " found");
	}
}
```

Question-3:
Write a program to convert a binary tree to a doubly linked list.
Input:



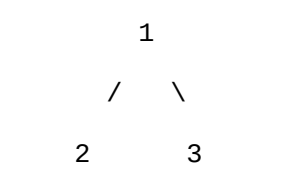
Output:
5 10 30 20 35

code:-

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

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

class BinaryTree
{
	Node root;
	Node head;
	static Node prev = null;
	void BinaryTree2DoubleLinkedList(Node root)
	{
		// Base case
		if (root == null)
			return;
		BinaryTree2DoubleLinkedList(root.left);
		if (prev == null)
			head = root;
		else
		{
			root.left = prev;
			prev.right = root;
		}
		prev = root;
		BinaryTree2DoubleLinkedList(root.right);
	}

	void printList(Node node)
	{
		while (node != null)
		{
			System.out.print(node.data + " ");
			node = node.right;
		}
	}
	public static void main(String[] args)
	{
		BinaryTree tree = new BinaryTree();
		tree.root = new Node(10);
		tree.root.left = new Node(12);
		tree.root.right = new Node(15);
		tree.root.left.left = new Node(25);
		tree.root.left.right = new Node(30);
		tree.root.right.left = new Node(36);
		tree.BinaryTree2DoubleLinkedList(tree.root);
		tree.printList(tree.head);
```

```
	}
}
```

Question-4:
Write a program to connect nodes at the same level.
Input:



Output:
1 → -1
2 → 3
3 → -1
4 → 5
5 → 6
6 → 7
7 → -1

code:-

```
import java.io.*;
import java.util.*;
class Node {
	int data;
	Node left, right, nextRight;

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

public class BinaryTree {
	Node root;
	void connect(Node p)
	{
		// initialize queue to hold nodes at same level
		Queue<Node> q = new LinkedList<>();
		q.add(root); // adding nodes to the queue
		Node temp = null; // initializing prev to null
		while (!q.isEmpty()) {
			int n = q.size();
			for (int i = 0; i < n; i++) {
				Node prev = temp;
				temp = q.poll();
				if (i > 0)
					prev.nextRight = temp;

				if (temp.left != null)
					q.add(temp.left);
```

```
                        if (temp.right != null)
                              q.add(temp.right);
                  }

                  // pointing last node of the nth level to null
                  temp.nextRight = null;
            }
      }

      public static void main(String args[])
      {
            BinaryTree tree = new BinaryTree();
            tree.root = new Node(10);
            tree.root.left = new Node(8);
            tree.root.right = new Node(2);
            tree.root.left.left = new Node(3);
            tree.connect(tree.root);
            System.out.println("Following are populated nextRight pointers in "+
"the tree"+ "(-1 is printed if there is no nextRight)");
            int a = tree.root.nextRight != null? tree.root.nextRight.data: -1;
            System.out.println("nextRight of " + tree.root.data+ " is " + a);
            int b = tree.root.left.nextRight != null?
tree.root.left.nextRight.data: -1;
            System.out.println("nextRight of "+ tree.root.left.data + " is "+ b);
            int c = tree.root.right.nextRight != null?
tree.root.right.nextRight.data: -1;
            System.out.println("nextRight of "+ tree.root.right.data + " is "+ c);
            int d = tree.root.left.left.nextRight != null?
tree.root.left.left.nextRight.data: -1;
            System.out.println("nextRight of "+ tree.root.left.left.data+ " is " +
d);
      }
}
```