Solution @(04/08/2023)

```
// A class to store a binary tree node
class Node
{
	int key;
	Node left = null, right = null;

	Node(int key) {
		this.key = key;
	}
}

class Main
{
	// Recursive function to check if two given binary trees are identical or not
	public static boolean isIdentical(Node x, Node y)
	{
		// if both trees are empty, return true
		if (x == null && y == null) {
			return true;
		}

		// if both trees are non-empty and the value of their root node matches,
		// recur for their left and right subtree
		return (x != null && y != null) && (x.key == y.key) &&
					isIdentical(x.left, y.left) &&
					isIdentical(x.right, y.right);
	}
```

```
	public static void main(String[] args)
	{
		// construct the first tree
		Node x = new Node(15);
		x.left = new Node(10);
		x.right = new Node(20);
		x.left.left = new Node(8);
		x.left.right = new Node(12);
		x.right.left = new Node(16);
		x.right.right = new Node(25);

		// construct the second tree
		Node y = new Node(15);
		y.left = new Node(10);
		y.right = new Node(20);
		y.left.left = new Node(8);
		y.left.right = new Node(12);
		y.right.left = new Node(16);
		y.right.right = new Node(25);

		if (isIdentical(x, y)) {
			System.out.println("The given binary trees are identical");
		}
		else {
			System.out.println("The given binary trees are not identical");
		}
	}
}
```

OR

```
class Solution {
    //When both the binary trees are null, then both the binary trees are same,
return true.
public boolean isSameTree(TreeNode p, TreeNode q) {
if (p == null && q == null)
return true;
//If one binary tree is null and the other is not null,then the two binary trees
are different, return false.
if (p == null || q == null)
return false;
//If the root nodes of both the binary trees has different values then the trees
are different, return false.
if (p.val != q.val)
return false;
//Now check the same for their left subtree and right subtree, if both return true
then the binary trees are same else they are different.
return isSameTree(p.left, q.left) && isSameTree(p.right, q.right);

}
}
```

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.

```
public class Solution {
    public int[] twoSum(int[] numbers, int target) {
        HashMap<Integer,Integer> indexMap = new HashMap<Integer,Integer>();
        for(int i = 0; i < numbers.length; i++){
            Integer requiredNum = (Integer)(target - numbers[i]);
            if(indexMap.containsKey(requiredNum)){
                int toReturn[] = {indexMap.get(requiredNum), i};
                return toReturn;
            }

            indexMap.put(numbers[i], i);
        }
        return null;
    }
}
```