Problem 3

Leet Code Problem to find Two sum

```
import java.util.Arrays;

class Solution {


   public int[] twoSum(int[] nums, int target)  // Getting the value for array and target value for main
   {
      for (int i = 0; i < nums.length; i++) {   //Loop to compare first value
         for (int j = i + 1; j < nums.length; j++) { // Loop to compare second value
            if (nums[i] + nums[j] == target) {  // If addition of both gets equal it will return i and j
               return new int[] {i, j};
            }
         }
      }
      return new int[] {};    // If its not equal it will return empty array
   }
}


class TwoSum{


   public static void main (String[] args)
   {
```

```
        int [] nums = {2,7,11,15}; // Array list to find the target
        int target = 18; // Target Value
        Solution obj = new Solution();   // Creating an object for Solution class so the it executes its constructor
        int []  result = obj.twoSum(nums,target); // Passing the array and target through argument and saving it in result
        System.out.println(Arrays.toString(result)); // Converting Array to String and printing it
    }

}
```

Leet Code Problem to find both trees are same

```
public class SameTree {
   // Method to check whether both tree are same
   public boolean isSameTree(TreeNode p, TreeNode q) {
      //To check if both trees are empty then they are same then return true
      if (p == null && q == null) {
         return true;
      }
      // If one of the tree is empty the they are not same then return false
      if (p == null || q == null) {
         return false;
      }
      // If the values of the nodes are not equal then return false
      if (p.val != q.val) {
         return false;
      }
      // Using recursion to check left node and right node
      return isSameTree(p.left, q.left) && isSameTree(p.right, q.right);
   }

   public static void main(String[] args) {
      // Creating two binary trees to test the isSameTree method
      TreeNode p = new TreeNode(1, new TreeNode(2), new TreeNode(3));
      TreeNode q = new TreeNode(1, new TreeNode(2), new TreeNode(3));
      SameTree obj = new SameTree();
```

```
        // Test the isSameTree method with the two binary trees
        System.out.println(obj.isSameTree(p, q));  // it will pass the aruguments to the isSameTree method

        // Create two more binary trees to test the isSameTree method
        p = new TreeNode(1, new TreeNode(2), null);
        q = new TreeNode(1, null, new TreeNode(2));
        System.out.println(obj.isSameTree(p, q));  // it will pass the aruguments to the isSameTree method

        p = new TreeNode(1, new TreeNode(2), new TreeNode(1));
        q = new TreeNode(1, new TreeNode(1), new TreeNode(2));
        System.out.println(obj.isSameTree(p, q)); // it will pass the aruguments to the isSameTree method
    }
}

//Here we are defining a TreeNode class to represent each node in a binary tree
class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode() {}
```

```
    TreeNode(int val) {
        this.val = val;
    }

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