DSA Practice

Date: 11/11/2024

Problems:

0-1 knapsack problem

Floor in sorted array

Check equal arrays

Palindrome linked list

Balanced tree check

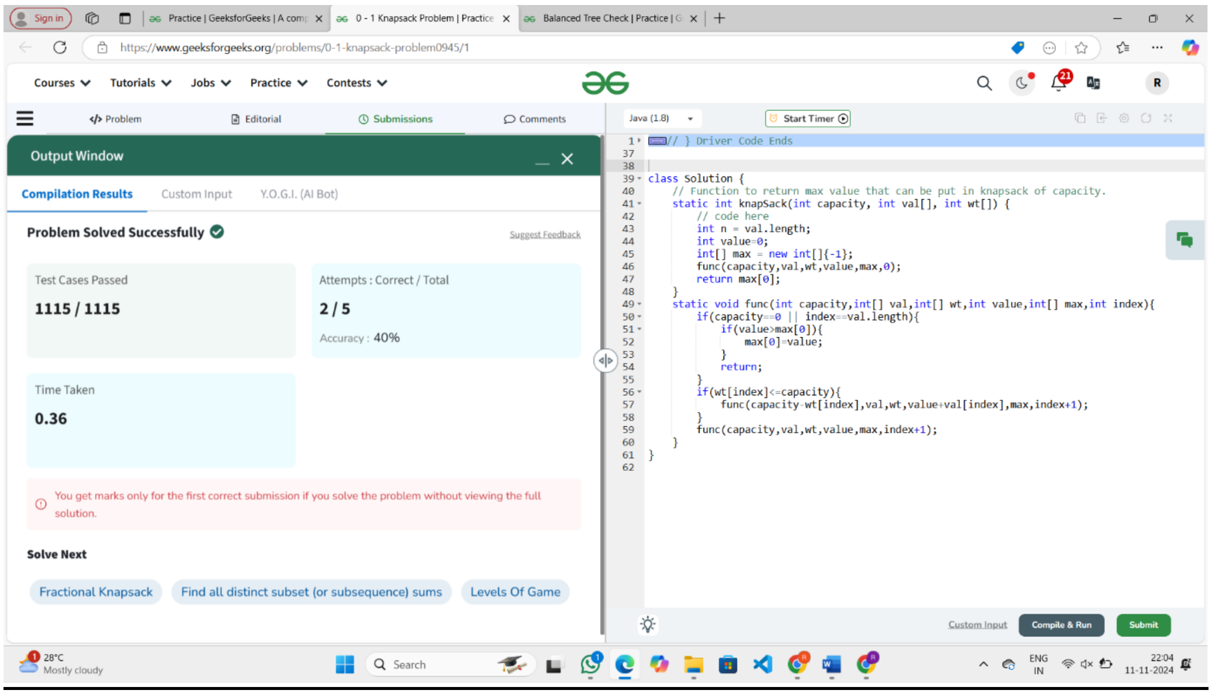
Triplet sum in array

Problem 1:

```
class Solution {
   // Function to return max value that can be put in knapsack of capacity.
   static int knapSack(int capacity, int val[], int wt[]) {
      // code here
      int n = val.length;
      int value=0;
      int[] max = new int[]{-1};
      func(capacity,val,wt,value,max,0);
      return max[0];
   }
   static void func(int capacity,int[] val,int[] wt,int value,int[] max,int index){
      if(capacity==0 || index==val.length){
         if(value>max[0]){
            max[0]=value;
         }
         return;
```

```
        }
        if(wt[index]<=capacity){
            func(capacity-wt[index],val,wt,value+val[index],max,index+1);
        }
        func(capacity,val,wt,value,max,index+1);
    }
}
```

Output:



https://www.geeksforgeeks.org/problems/0-1-knapsack-problem0945/1

Output Window

Compilation Results | Custom Input | Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed
1115 / 1115

Attempts : Correct / Total
2 / 5
Accuracy : 40%

Time Taken
0.36

You get marks only for the first correct submission if you solve the problem without viewing the full solution.

Solve Next

Fractional Knapsack | Find all distinct subset (or subsequence) sums | Levels Of Game

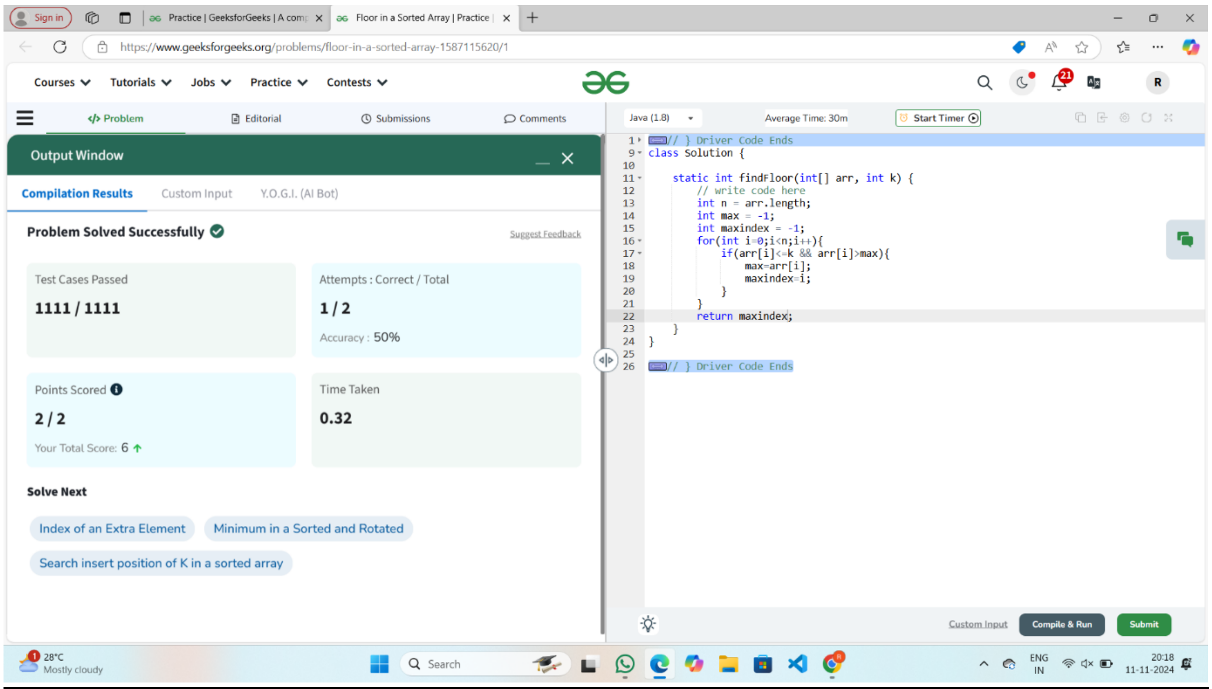
Java (1.8)

```java
// } Driver Code Ends

class Solution {
    // Function to return max value that can be put in knapsack of capacity.
    static int knapSack(int capacity, int val[], int wt[]) {
        // code here
        int n = val.length;
        int value=0;
        int[] max = new int[]{-1};
        func(capacity,val,wt,value,max,0);
        return max[0];
    }
    static void func(int capacity,int[] val,int[] wt,int value,int[] max,int index){
        if(capacity==0 || index==val.length){
            if(value>max[0]){
                max[0]=value;
            }
            return;
        }
        if(wt[index]<=capacity){
            func(capacity-wt[index],val,wt,value+val[index],max,index+1);
        }
        func(capacity,val,wt,value,max,index+1);
    }
}
```

Problem 2:

```
Class Solution{
    static int findFloor(int[] arr, int k) {
        // write code here
        int n = arr.length;
        int max = -1;
        int maxindex = -1;
        for(int i=0;i<n;i++){
            if(arr[i]<=k && arr[i]>max){
                max=arr[i];
                maxindex=i;
            }
        }
        return maxindex;
    }
}
```

Output:



Floor in a Sorted Array | Practice
https://www.geeksforgeeks.org/problems/floor-in-a-sorted-array-1587115620/1

Output Window

Compilation Results

Problem Solved Successfully

Test Cases Passed
1111 / 1111

Attempts : Correct / Total
1 / 2
Accuracy : 50%

Points Scored
2 / 2
Your Total Score: 6

Time Taken
0.32

Solve Next
Index of an Extra Element
Minimum in a Sorted and Rotated
Search insert position of K in a sorted array

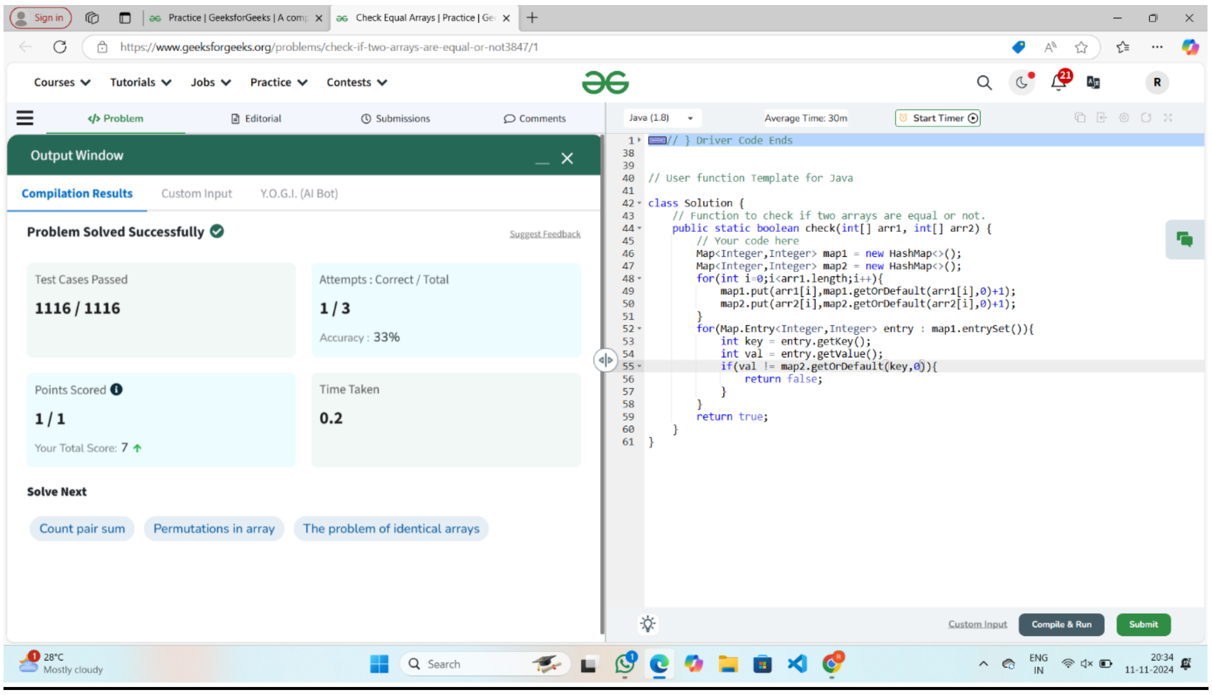
Java (1.8)

```
// } Driver Code Ends
class Solution {

    static int findFloor(int[] arr, int k) {
        // write code here
        int n = arr.length;
        int max = -1;
        int maxindex = -1;
        for(int i=0;i<n;i++){
            if(arr[i]<=k && arr[i]>max){
                max=arr[i];
                maxindex=i;
            }
        }
        return maxindex;
    }
}

// } Driver Code Ends
```

Problem 3:

```
class Solution {
    // Function to check if two arrays are equal or not.
    public static boolean check(int[] arr1, int[] arr2) {
        // Your code here
        Map<Integer,Integer> map1 = new HashMap<>();
        Map<Integer,Integer> map2 = new HashMap<>();
        for(int i=0;i<arr1.length;i++){
            map1.put(arr1[i],map1.getOrDefault(arr1[i],0)+1);
            map2.put(arr2[i],map2.getOrDefault(arr2[i],0)+1);
        }
        for(Map.Entry<Integer,Integer> entry : map1.entrySet()){
            int key = entry.getKey();
            int val = entry.getValue();
            if(val != map2.getOrDefault(key,0)){
                return false;
            }
        }
        return true;
    }
}
```

Output:



https://www.geeksforgeeks.org/problems/check-if-two-arrays-are-equal-or-not3847/1

Output Window

Compilation Results | Custom Input | Y.O.G.I. (AI Bot)

Problem Solved Successfully

Test Cases Passed
1116 / 1116

Attempts : Correct / Total
1 / 3
Accuracy : 33%

Points Scored
1 / 1
Your Total Score: 7

Time Taken
0.2

Solve Next

Count pair sum | Permutations in array | The problem of identical arrays

Java (1.8) Average Time: 30m

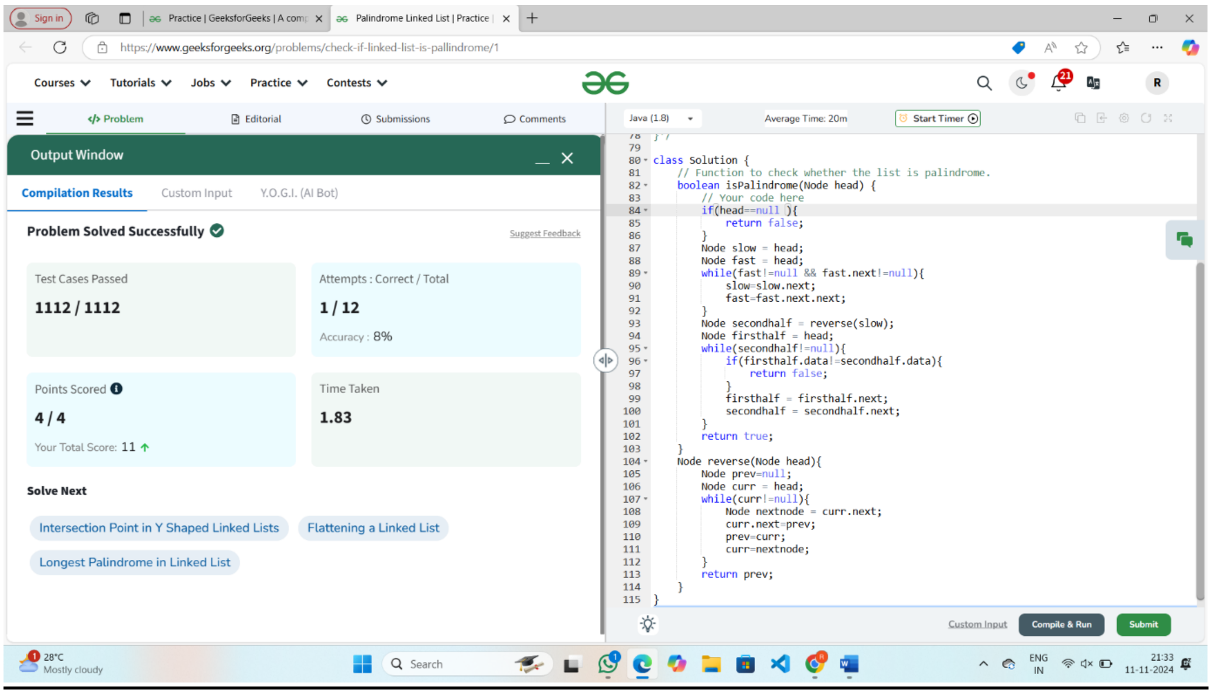
```
// } Driver Code Ends


// User function Template for Java

class Solution {
    // Function to check if two arrays are equal or not.
    public static boolean check(int[] arr1, int[] arr2) {
        // Your code here
        Map<Integer,Integer> map1 = new HashMap<>();
        Map<Integer,Integer> map2 = new HashMap<>();
        for(int i=0;i<arr1.length;i++){
            map1.put(arr1[i],map1.getOrDefault(arr1[i],0)+1);
            map2.put(arr2[i],map2.getOrDefault(arr2[i],0)+1);
        }
        for(Map.Entry<Integer,Integer> entry : map1.entrySet()){
            int key = entry.getKey();
            int val = entry.getValue();
            if(val != map2.getOrDefault(key,0)){
                return false;
            }
        }
        return true;
    }
}
```

Problem 4:

```
class Solution {
  // Function to check whether the list is palindrome.
  boolean isPalindrome(Node head) {
    // Your code here
    if(head==null ){
      return false;
    }
    Node slow = head;
    Node fast = head;
    while(fast!=null && fast.next!=null){
      slow=slow.next;
      fast=fast.next.next;
    }
    Node secondhalf = reverse(slow);
    Node firsthalf = head;
    while(secondhalf!=null){
      if(firsthalf.data!=secondhalf.data){
        return false;
      }
      firsthalf = firsthalf.next;
      secondhalf = secondhalf.next;
    }
    return true;
  }
  Node reverse(Node head){
    Node prev=null;
```

```
        Node curr = head;
        while(curr!=null){
            Node nextnode = curr.next;
            curr.next=prev;
            prev=curr;
            curr=nextnode;
        }
        return prev;
    }
}
```

<u>Output:</u>



Problem Solved Successfully

Test Cases Passed
1112 / 1112

Attempts : Correct / Total
1 / 12
Accuracy : 8%

Points Scored
4 / 4
Your Total Score: 11

Time Taken
1.83

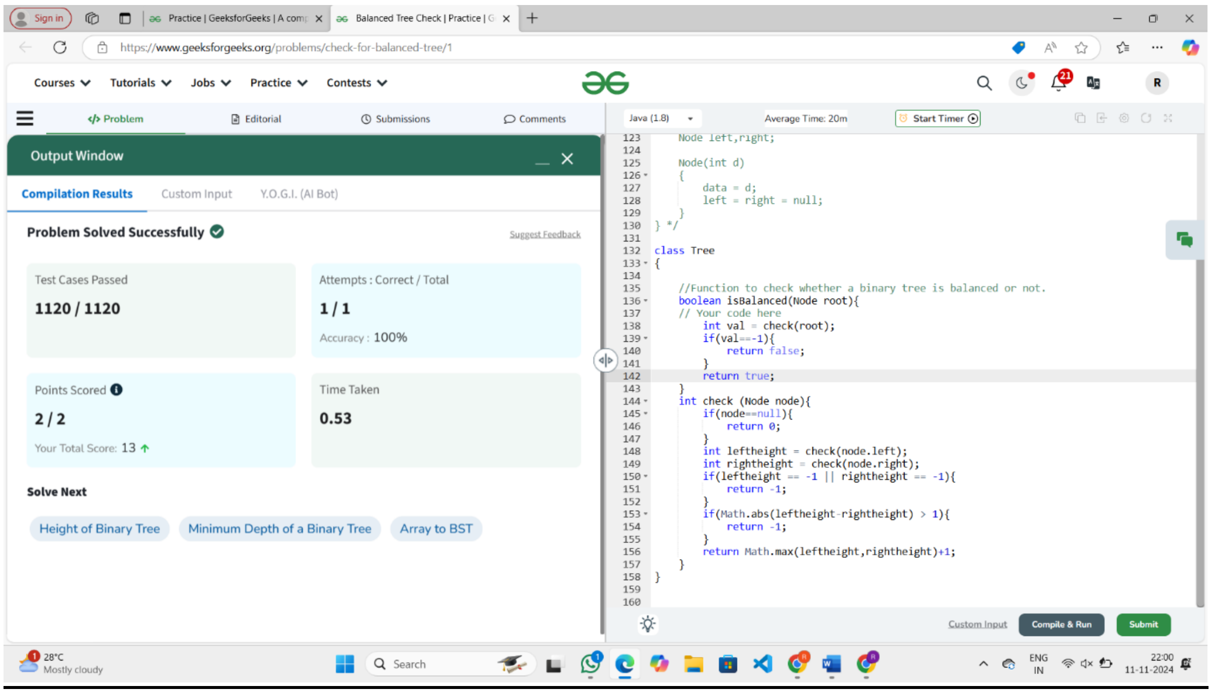
Solve Next
Intersection Point in Y Shaped Linked Lists
Flattening a Linked List
Longest Palindrome in Linked List

```
}*/

class Solution {
    // Function to check whether the list is palindrome.
    boolean isPalindrome(Node head) {
        // Your code here
        if(head==null ){
            return false;
        }
        Node slow = head;
        Node fast = head;
        while(fast!=null && fast.next!=null){
            slow=slow.next;
            fast=fast.next.next;
        }
        Node secondhalf = reverse(slow);
        Node firsthalf = head;
        while(secondhalf!=null){
            if(firsthalf.data!=secondhalf.data){
                return false;
            }
            firsthalf = firsthalf.next;
            secondhalf = secondhalf.next;
        }
        return true;
    }
    Node reverse(Node head){
        Node prev=null;
        Node curr = head;
        while(curr!=null){
            Node nextnode = curr.next;
            curr.next=prev;
            prev=curr;
            curr=nextnode;
        }
        return prev;
    }
}
```

<u>Problem 5:</u>

```
class Tree
{

    //Function to check whether a binary tree is balanced or not.
    boolean isBalanced(Node root){
        // Your code here
        int val = check(root);
            if(val==-1){
                return false;
            }
        return true;
    }
    int check (Node node){
        if(node==null){
            return 0;
        }
        int leftheight = check(node.left);
        int rightheight = check(node.right);
        if(leftheight == -1 || rightheight == -1){
            return -1;
        }
        if(Math.abs(leftheight-rightheight) > 1){
            return -1;
        }
        return Math.max(leftheight,rightheight)+1;
    }
}
```

Output:



Output Window

Compilation Results

Problem Solved Successfully

Test Cases Passed
1120 / 1120

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
2 / 2
Your Total Score: 13

Time Taken
0.53

Solve Next

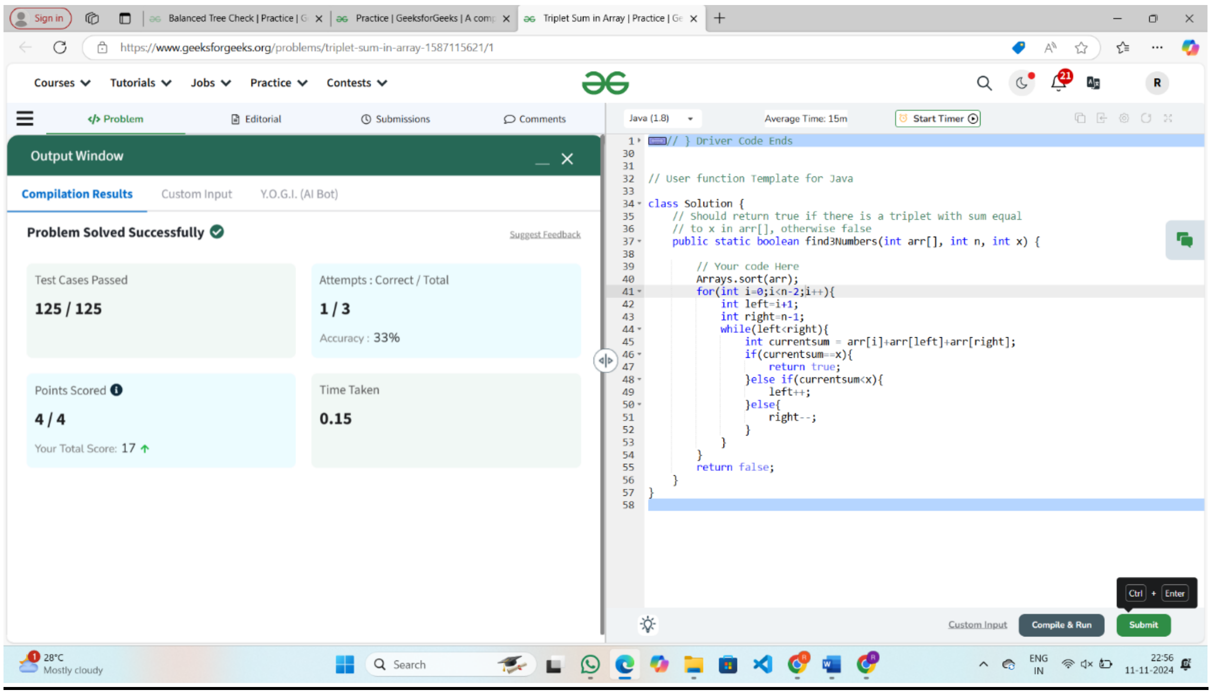
Height of Binary Tree · Minimum Depth of a Binary Tree · Array to BST

```
    Node left,right;

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

class Tree
{

    //Function to check whether a binary tree is balanced or not.
    boolean isBalanced(Node root){
    // Your code here
        int val = check(root);
        if(val==-1){
            return false;
        }
        return true;
    }
    int check (Node node){
        if(node==null){
            return 0;
        }
        int leftheight = check(node.left);
        int rightheight = check(node.right);
        if(leftheight == -1 || rightheight == -1){
            return -1;
        }
        if(Math.abs(leftheight-rightheight) > 1){
            return -1;
        }
        return Math.max(leftheight,rightheight)+1;
    }
}
```

<u>Problem 6:</u>

```
class Solution {
   // Should return true if there is a triplet with sum equal
   // to x in arr[], otherwise false
   public static boolean find3Numbers(int arr[], int n, int x) {

       // Your code Here
       Arrays.sort(arr);
       for(int i=0;i<n-2;i++){
           int left=i+1;
           int right=n-1;
           while(left<right){
               int currentsum = arr[i]+arr[left]+arr[right];
               if(currentsum==x){
                   return true;
               }else if(currentsum<x){
                    left++;
                }else{
                    right--;
                }
            }
        }
        return false;
    }
}
```

Output:



Output Window

Compilation Results | Custom Input | Y.O.G.I. (AI Bot)

Problem Solved Successfully

Test Cases Passed
125 / 125

Attempts : Correct / Total
1 / 3
Accuracy : 33%

Points Scored
4 / 4
Your Total Score: 17

Time Taken
0.15

```
// } Driver Code Ends


// User function Template for Java

class Solution {
    // Should return true if there is a triplet with sum equal
    // to x in arr[], otherwise false
    public static boolean find3Numbers(int arr[], int n, int x) {

        // Your code Here
        Arrays.sort(arr);
        for(int i=0;i<n-2;i++){
            int left=i+1;
            int right=n-1;
            while(left<right){
                int currentsum = arr[i]+arr[left]+arr[right];
                if(currentsum==x){
                    return true;
                }else if(currentsum<x){
                    left++;
                }else{
                    right--;
                }
            }
        }
        return false;
    }
}
```