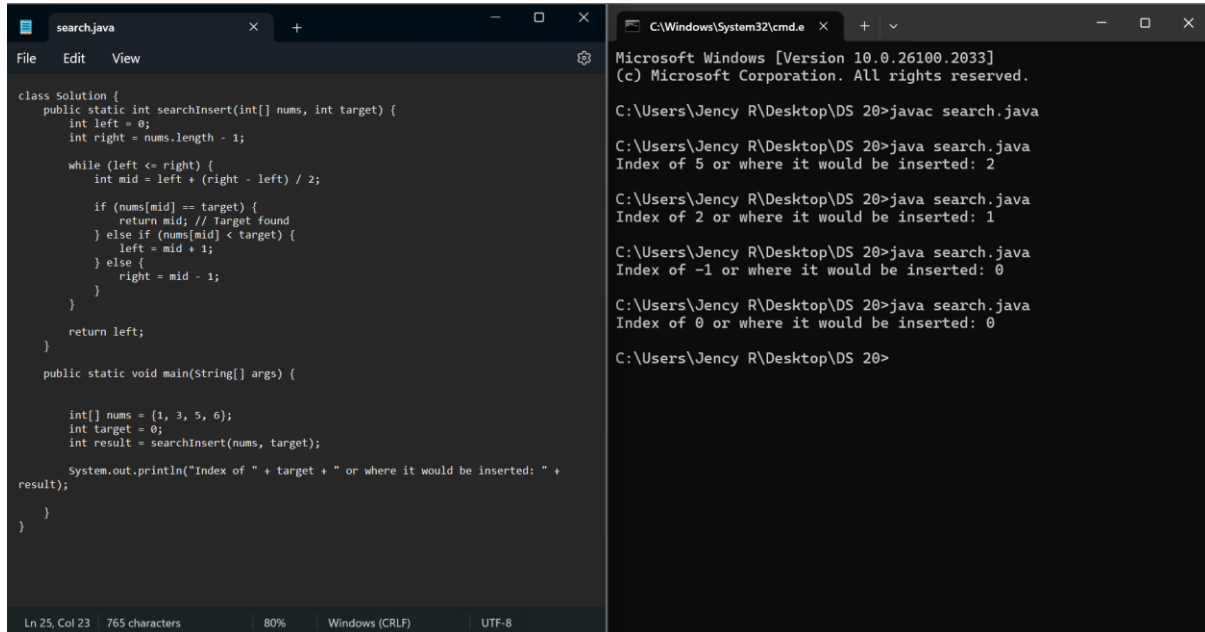


DSA PRACTICE CODING QUESTION – 2

1. Floor in sorted array



```
search.java
class Solution {
    public static int searchInsert(int[] nums, int target) {
        int left = 0;
        int right = nums.length - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;

            if (nums[mid] == target) {
                return mid; // Target found
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }

        return left;
    }

    public static void main(String[] args) {

        int[] nums = {1, 3, 5, 6};
        int target = 0;
        int result = searchInsert(nums, target);

        System.out.println("Index of " + target + " or where it would be inserted: " + result);
    }
}

C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.26100.2033]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Jency R\Desktop\DS 20>javac search.java

C:\Users\Jency R\Desktop\DS 20>java search.java
Index of 5 or where it would be inserted: 2

C:\Users\Jency R\Desktop\DS 20>java search.java
Index of 2 or where it would be inserted: 1

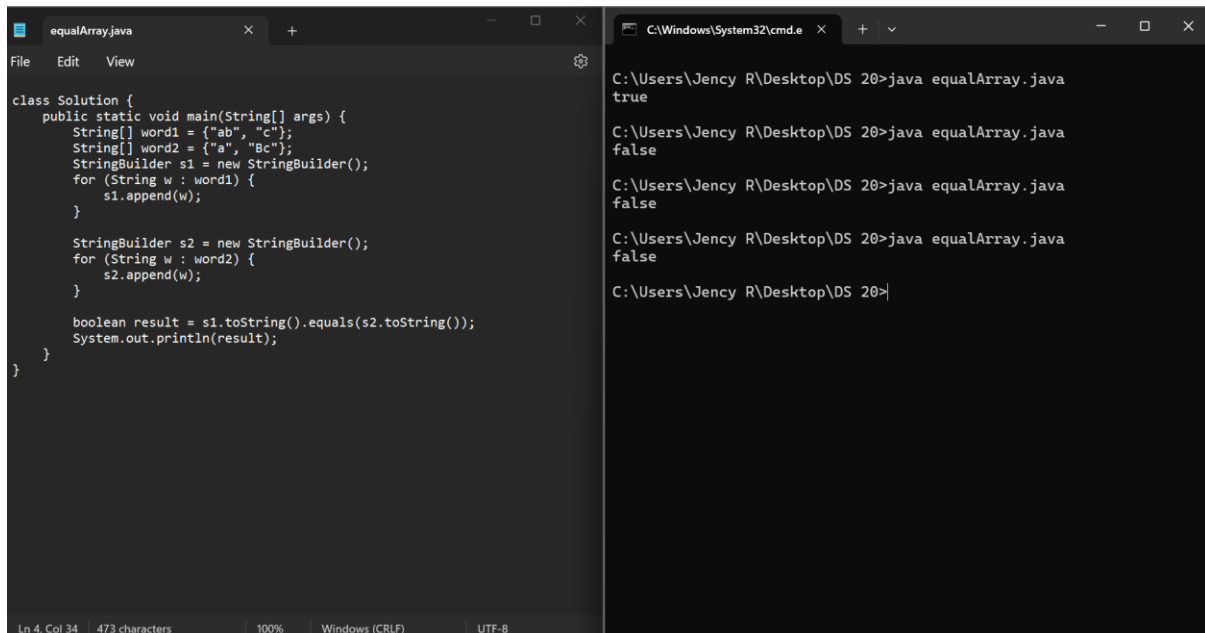
C:\Users\Jency R\Desktop\DS 20>java search.java
Index of -1 or where it would be inserted: 0

C:\Users\Jency R\Desktop\DS 20>java search.java
Index of 0 or where it would be inserted: 0

C:\Users\Jency R\Desktop\DS 20>
```

Time Complexity: $O(\log n)$, **Space Complexity:** $O(1)$

2. Check equal arrays



```
equalArray.java
class Solution {
    public static void main(String[] args) {
        String[] word1 = {"ab", "c"};
        String[] word2 = {"a", "Bc"};
        StringBuilder s1 = new StringBuilder();
        for (String w : word1) {
            s1.append(w);
        }

        StringBuilder s2 = new StringBuilder();
        for (String w : word2) {
            s2.append(w);
        }

        boolean result = s1.toString().equals(s2.toString());
        System.out.println(result);
    }
}

C:\Users\Jency R\Desktop\DS 20>java equalArray.java
true

C:\Users\Jency R\Desktop\DS 20>java equalArray.java
false

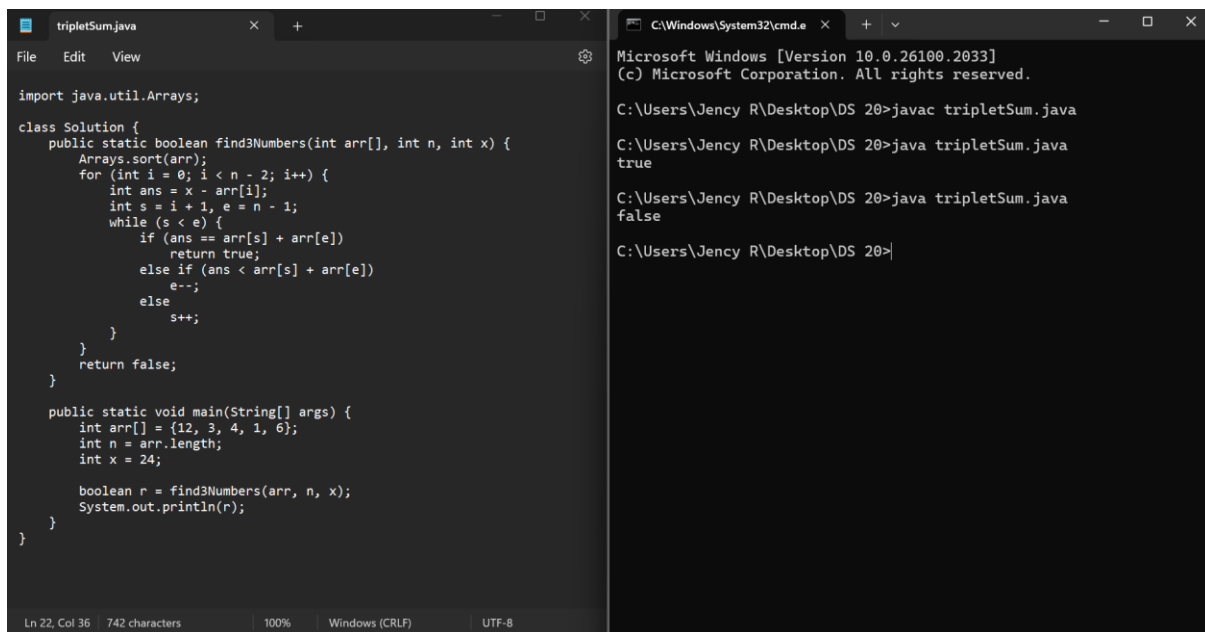
C:\Users\Jency R\Desktop\DS 20>java equalArray.java
false

C:\Users\Jency R\Desktop\DS 20>java equalArray.java
false

C:\Users\Jency R\Desktop\DS 20>
```

Time Complexity: $O(n)$, **Space Complexity:** $O(1)$

3. Triplet sum in array



The screenshot shows a Java IDE with a file named `tripletSum.java` and a Windows command prompt. The IDE contains the following code:

```
import java.util.Arrays;

class Solution {
    public static boolean find3Numbers(int arr[], int n, int x) {
        Arrays.sort(arr);
        for (int i = 0; i < n - 2; i++) {
            int ans = x - arr[i];
            int s = i + 1, e = n - 1;
            while (s < e) {
                if (ans == arr[s] + arr[e])
                    return true;
                else if (ans < arr[s] + arr[e])
                    e--;
                else
                    s++;
            }
        }
        return false;
    }

    public static void main(String[] args) {
        int arr[] = {12, 3, 4, 1, 6};
        int n = arr.length;
        int x = 24;

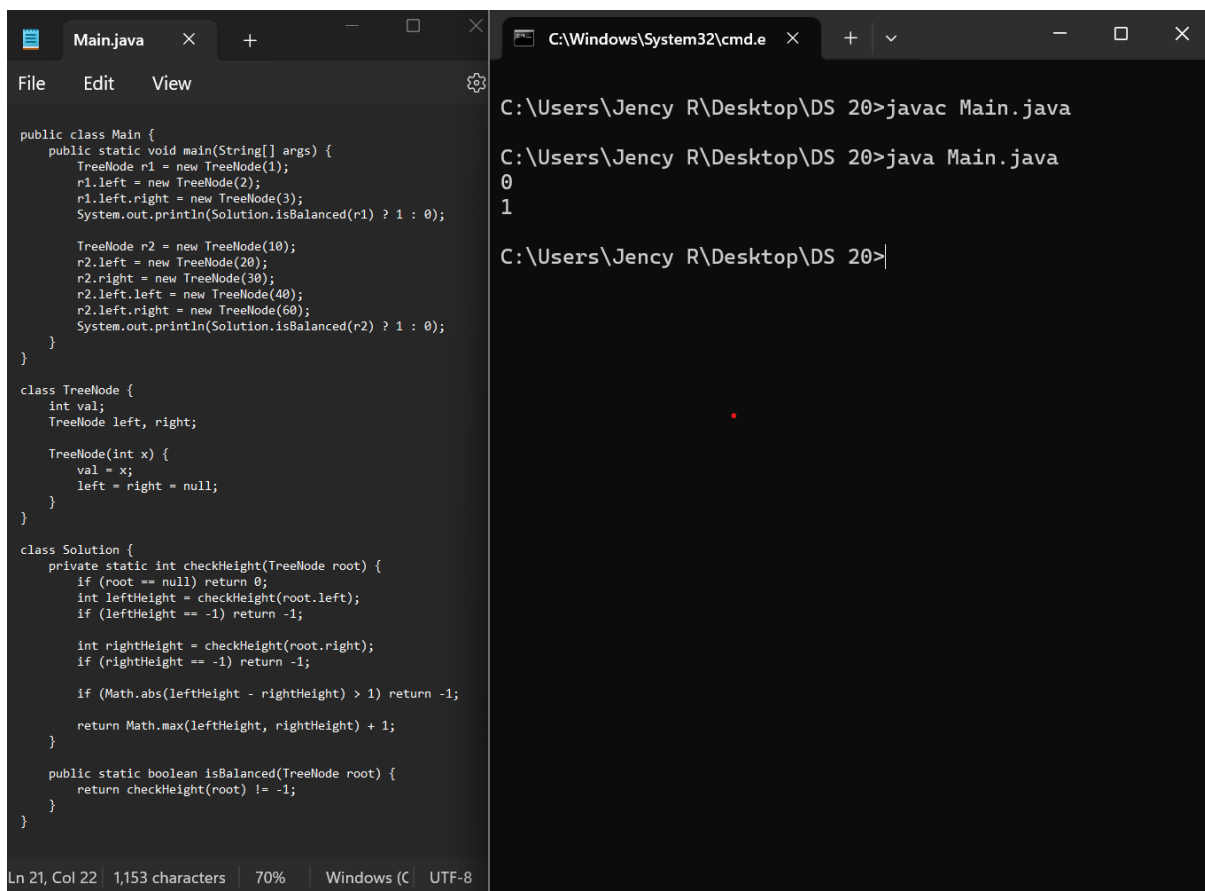
        boolean r = find3Numbers(arr, n, x);
        System.out.println(r);
    }
}
```

The command prompt shows the following commands and output:

```
C:\Users\Jency R\Desktop\DS 20>javac tripletSum.java
C:\Users\Jency R\Desktop\DS 20>java tripletSum.java
true
C:\Users\Jency R\Desktop\DS 20>java tripletSum.java
false
C:\Users\Jency R\Desktop\DS 20>|
```

Time Complexity: $O(n^2)$, Space Complexity: $O(1)$

4. Balanced Tree Check



The screenshot shows a Java IDE with a file named `Main.java` and a Windows command prompt. The IDE contains the following code:

```
public class Main {
    public static void main(String[] args) {
        TreeNode r1 = new TreeNode(1);
        r1.left = new TreeNode(2);
        r1.left.right = new TreeNode(3);
        System.out.println(Solution.isBalanced(r1) ? 1 : 0);

        TreeNode r2 = new TreeNode(10);
        r2.left = new TreeNode(20);
        r2.right = new TreeNode(30);
        r2.left.left = new TreeNode(40);
        r2.left.right = new TreeNode(60);
        System.out.println(Solution.isBalanced(r2) ? 1 : 0);
    }
}

class TreeNode {
    int val;
    TreeNode left, right;

    TreeNode(int x) {
        val = x;
        left = right = null;
    }
}

class Solution {
    private static int checkHeight(TreeNode root) {
        if (root == null) return 0;
        int leftHeight = checkHeight(root.left);
        if (leftHeight == -1) return -1;

        int rightHeight = checkHeight(root.right);
        if (rightHeight == -1) return -1;

        if (Math.abs(leftHeight - rightHeight) > 1) return -1;

        return Math.max(leftHeight, rightHeight) + 1;
    }

    public static boolean isBalanced(TreeNode root) {
        return checkHeight(root) != -1;
    }
}
```

The command prompt shows the following commands and output:

```
C:\Users\Jency R\Desktop\DS 20>javac Main.java
C:\Users\Jency R\Desktop\DS 20>java Main.java
0
1
C:\Users\Jency R\Desktop\DS 20>|
```

Time Complexity: $O(n)$, Space Complexity: $O(h)$

5. Palindrome LinkedList

The screenshot shows a coding platform interface for the 'Palindrome LinkedList' problem. The left sidebar displays the 'Output Window' with 'Compilation Results' for 'Y.O.G.I. (AI Bot)'. It indicates 'Problem Solved Successfully' with a green checkmark. Statistics shown include: Test Cases Passed (1112 / 1112), Attempts: Correct / Total (1 / 2), Accuracy (50%), Points Scored (4 / 4), and Time Taken (1.93). Below these, it suggests solving the next problem, 'Intersection in Y Shaped Lists'. The main editor area shows the Java code for the solution, which uses a stack to check if the linked list is a palindrome. The code is as follows:

```
69 {
70     int data;
71     Node next;
72
73     Node(int d)
74     {
75         data = d;
76         next = null;
77     }
78 }
79
80 class Solution {
81     // Function to check whether the list is palindrome.
82     boolean isPalindrome(Node head) {
83         // Your code here
84         Stack<Integer> s = new Stack();
85         Node curr = head;
86         while(curr != null){
87             s.push(curr.data);
88             curr = curr.next;
89         }
90         curr = head;
91         while(curr != null){
92             if(curr.data != s.pop()){
93                 return false;
94             }
95             curr = curr.next;
96         }
97         return true;
98     }
99 }
100 }
```

Time Complexity: $O(n)$, Space Complexity: $O(1)$