Solution@20-12-23

561. Array Partition

```
import java.util.Arrays;

import java.util.Scanner;


public class ArrayPartition {

  public static int arrayPairSum(int[] nums) {

    // Sort the array in ascending order

    Arrays.sort(nums);


    int sum = 0;


    // Sum every alternate element

    for (int i = 0; i < nums.length; i += 2) {

      sum += nums[i];

    }


    return sum;

  }


  public static void main(String[] args) {

    // Read input array length from the user

    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter the length of the array (even number): ");

    int n = scanner.nextInt();


    // Check if the length is even

    if (n % 2 != 0) {

      System.out.println("Please enter an even number for the length of the array.");

      return;

    }
```

```
        int[] nums = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }

        // Call the arrayPairSum method with user input
        int result = arrayPairSum(nums);
        System.out.println("Maximized sum: " + result);

        // Close the scanner
        scanner.close();
    }
}
```

Ouput:

```
PS C:\Users\Ajeet\Desktop\java> java ArrayPartition
Enter the length of the array (even number): 6
Enter the elements of the array:
1 2 3 4 5 6
Maximized sum: 9
```

567. Permutation in String

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class PermutationInString {
    public static boolean checkInclusion(String s1, String s2) {
        if (s1.length() > s2.length()) {
            return false;
        }

        Map<Character, Integer> s1Map = new HashMap<>();
        Map<Character, Integer> windowMap = new HashMap<>();

        // Initialize frequency map for s1
        for (char ch : s1.toCharArray()) {
            s1Map.put(ch, s1Map.getOrDefault(ch, 0) + 1);
        }

        int windowSize = s1.length();

        // Initialize frequency map for the first window in s2
        for (int i = 0; i < windowSize; i++) {
            char ch = s2.charAt(i);
            windowMap.put(ch, windowMap.getOrDefault(ch, 0) + 1);
        }

        // Iterate through the rest of s2
        for (int i = windowSize; i < s2.length(); i++) {
            if (windowMapsAreEqual(s1Map, windowMap)) {
                return true;
            }

            // Update window frequency map by removing the leftmost character and adding the rightmost
character
            char leftChar = s2.charAt(i - windowSize);
            windowMap.put(leftChar, windowMap.get(leftChar) - 1);
            if (windowMap.get(leftChar) == 0) {
                windowMap.remove(leftChar);
            }

            char rightChar = s2.charAt(i);
            windowMap.put(rightChar, windowMap.getOrDefault(rightChar, 0) + 1);
        }

        // Check the last window
        return windowMapsAreEqual(s1Map, windowMap);
    }

    private static boolean windowMapsAreEqual(Map<Character, Integer> map1, Map<Character,
Integer> map2) {
        for (char key : map1.keySet()) {
            if (!map2.containsKey(key) || !map2.get(key).equals(map1.get(key))) {
                return false;
            }
        }
        return true;
    }

    public static void main(String[] args) {
        // Read input strings from the user
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the first string (s1): ");
        String s1 = scanner.nextLine();
        System.out.print("Enter the second string (s2): ");
        String s2 = scanner.nextLine();

        // Call the checkInclusion method with user input
        boolean result = checkInclusion(s1, s2);
        System.out.println("Does s2 contain a permutation of s1? " + result);
        // Close the scanner
        scanner.close();
    }
}
```

Output:-

```
PS C:\Users\Ajeet\Desktop\java> javac PermutationInString.java
PS C:\Users\Ajeet\Desktop\java> java PermutationInString
Enter the first string (s1): ab
Enter the second string (s2): eidbaooo
Does s2 contain a permutation of s1? True
```

572. Subtree of Another Tree

```
import java.util.Scanner;

class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int x) {
        val = x;
    }
}

public class SubtreeOfAnotherTree {
    public static boolean isSubtree(TreeNode root, TreeNode subRoot) {
        if (root == null) {
            return false;
        }

        if (isSameTree(root, subRoot)) {
            return true;
        }

        return isSubtree(root.left, subRoot) || isSubtree(root.right, subRoot);
    }

    private static boolean isSameTree(TreeNode p, TreeNode q) {
        if (p == null && q == null) {
            return true;
        }
        if (p == null || q == null) {
            return false;
        }
        return (p.val == q.val) && isSameTree(p.left, q.left) && isSameTree(p.right, q.right);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input for Tree 1
        System.out.println("Enter values for Tree 1:");
        TreeNode root = readTree(scanner);

        // Input for Tree 2
        System.out.println("Enter values for Tree 2:");
        TreeNode subRoot = readTree(scanner);

        // Check if subRoot is a subtree of root
```

```
        boolean result = isSubtree(root, subRoot);
        System.out.println("Is subRoot a subtree of root? " + result);

        // Close the scanner
        scanner.close();
    }

    private static TreeNode readTree(Scanner scanner) {
        System.out.print("Enter root value: ");
        int rootValue = scanner.nextInt();
        TreeNode root = new TreeNode(rootValue);

        readChildNodes(scanner, root);

        return root;
    }

    private static void readChildNodes(Scanner scanner, TreeNode parent) {
        System.out.print("Enter left child value of " + parent.val + " (enter -1 if no left child): ");
        int leftValue = scanner.nextInt();
        if (leftValue != -1) {
            parent.left = new TreeNode(leftValue);
            readChildNodes(scanner, parent.left);
        }

        System.out.print("Enter right child value of " + parent.val + " (enter -1 if no right child): ");
        int rightValue = scanner.nextInt();
        if (rightValue != -1) {
            parent.right = new TreeNode(rightValue);
            readChildNodes(scanner, parent.right);
        }
    }
}
```

Output

```
PS C:\Users\Ajeet\Desktop\java> javac SubtreeOfAnotherTree.java
PS C:\Users\Ajeet\Desktop\java> java SubtreeOfAnotherTree
Enter values for Tree 1:
Root value: 2
Left child of 2 (enter -1 if no left child): 1
Left child of 1 (enter -1 if no left child): 1
Left child of 1 (enter -1 if no left child):
2
Left child of 2 (enter -1 if no left child): -1
Right child of 2 (enter -1 if no right child): 2
Left child of 2 (enter -1 if no left child): 3
Left child of 3 (enter -1 if no left child): -1
Right child of 3 (enter -1 if no right child): -1
Right child of 2 (enter -1 if no right child): -1
Right child of 1 (enter -1 if no right child): -1
Right child of 1 (enter -1 if no right child): -1
Right child of 2 (enter -1 if no right child): -1
Enter values for Tree 2:
Root value: 3
Left child of 3 (enter -1 if no left child): 2
Left child of 2 (enter -1 if no left child): 1
Left child of 1 (enter -1 if no left child): 2
Left child of 2 (enter -1 if no left child): 3
Left child of 3 (enter -1 if no left child): 4
Left child of 4 (enter -1 if no left child): 0
Left child of 0 (enter -1 if no left child): -1
Right child of 0 (enter -1 if no right child): -1
Right child of 4 (enter -1 if no right child): -1
Right child of 3 (enter -1 if no right child): -1
Right child of 2 (enter -1 if no right child): -1
Right child of 1 (enter -1 if no right child): -1
Right child of 2 (enter -1 if no right child): -1
Right child of 3 (enter -1 if no right child): -1
Is subRoot a subtree of root? false
PS C:\Users\Ajeet\Desktop\java>
```

583. Delete Operation for Two Strings

```
import java.util.Scanner;

public class DeleteOperationForTwoStrings {
    public static int minDistance(String word1, String word2) {
        int m = word1.length();
        int n = word2.length();

        int[][] dp = new int[m + 1][n + 1];

        for (int i = 0; i <= m; i++) {
            for (int j = 0; j <= n; j++) {
                if (i == 0 || j == 0) {
                    dp[i][j] = i + j;
                } else if (word1.charAt(i - 1) == word2.charAt(j - 1)) {
                    dp[i][j] = dp[i - 1][j - 1];
                } else {
                    dp[i][j] = 1 + Math.min(dp[i - 1][j], dp[i][j - 1]);
                }
            }
        }
        return dp[m][n];
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input for word1
        System.out.print("Enter the first word: ");
        String word1 = scanner.nextLine();

        // Input for word2
        System.out.print("Enter the second word: ");
        String word2 = scanner.nextLine();

        // Calculate and display the minimum number of steps
        int result = minDistance(word1, word2);
        System.out.println("Minimum number of steps: " + result);

        // Close the scanner
        scanner.close();
    }
}
```

output

```
PS C:\Users\Ajeet\Desktop\java> javac DeleteOperationForTwoStrings.java
PS C:\Users\Ajeet\Desktop\java> java DeleteOperationForTwoStrings
Enter the first word: sea
Enter the second word: eat
Minimum number of steps: 2
```