**DAY-6**

**Quiz-1**

**Easy:**

Given the list of array return array in which each element is the product of other elem ent except ith element (try to do it without division operation)
input: [1,2,3,4]
output:[24,12,8,6]

**CODE:**

```
import java.util.Scanner;
public class ProductExceptSelf {
    public static int[] productExceptSelf(int[] nums) {
        int n = nums.length;
        int[] result = new int[n];
        int[] leftProducts = new int[n];
        leftProducts[0] = 1;
        for (int i = 1; i < n; i++) {
            leftProducts[i] = leftProducts[i - 1] * nums[i - 1];
        }
        int rightProduct = 1;
        for (int i = n - 1; i >= 0; i--) {
            result[i] = leftProducts[i] * rightProduct;
            rightProduct *= nums[i];
        }

        return result;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
        int[] input = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            input[i] = scanner.nextInt();
        }
        int[] output = productExceptSelf(input);
        System.out.print("Output: [");
        for (int i = 0; i < output.length; i++) {
            System.out.print(output[i]);
            if (i < output.length - 1) {
                System.out.print(", ");
            }
        }
```

```
        System.out.println("]");
        scanner.close();
    }
}
```

**OUTPUT:**

```
Enter the number of elements in the array: 4
Enter the elements of the array:
1 2 3 4
Output: [24, 12, 8, 6]
```

**Medium:**

Given an array list return all possible permutations Input: nums = [1,4,3]
Output: [[1,4,3],[1,3,4],[4,1,3],[4,3,1],[3,1,4],[3,4,1]]

**CODE:**

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class Permutations {
    public static List<List<Integer>> permute(int[] nums) {
        List<List<Integer>> result = new ArrayList<>();
        List<Integer> currentPermutation = new ArrayList<>();
        boolean[] used = new boolean[nums.length];
        generatePermutations(nums, used, currentPermutation, result);
        return result;
    }
    private static void generatePermutations(int[] nums, boolean[] used, List<Integer>
currentPermutation, List<List<Integer>> result) {
        if (currentPermutation.size() == nums.length) {
            result.add(new ArrayList<>(currentPermutation));
            return;
        }
        for (int i = 0; i < nums.length; i++) {
            if (!used[i]) {
                used[i] = true;
                currentPermutation.add(nums[i]);
                generatePermutations(nums, used, currentPermutation, result);
                currentPermutation.remove(currentPermutation.size() - 1);
                used[i] = false;
            }
        }
    }
```

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
        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();
        }
        List<List<Integer>> permutations = permute(nums);
        System.out.println("Output: " + permutations);

        scanner.close();
    }
}
```

**OUTPUT:**

```
Enter the number of elements in the array: 3
Enter the elements of the array:1 4 3
Output: [[1, 4, 3], [1, 3, 4], [4, 1, 3], [4, 3, 1], [3, 1, 4], [3, 4, 1]]
```

**Hard:**

Return all the clubbed words
Input:
words =["mat","mate","matbellmates","bell","bellmatesbell","butterribbon","butter","ribbon"] Output: ["matbellmates","bellmatesbell","butterribbon"]

**CODE:**

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class ClubbedWords {
    static class TrieNode {
        java.util.Map<Character, TrieNode> children = new java.util.HashMap<>();
        boolean isEndOfWord;
    }
    static class Trie {
        TrieNode root = new TrieNode();
        void insert(String word) {
            TrieNode node = root;
            for (char ch : word.toCharArray()) {
                node.children.putIfAbsent(ch, new TrieNode());
                node = node.children.get(ch);
            }
            node.isEndOfWord = true;
        }
```

```
    boolean isClubbedWord(String word) {
        TrieNode node = root;
        for (char ch : word.toCharArray()) {
            if (!node.children.containsKey(ch)) {
                return false;
            }
            node = node.children.get(ch);
        }
        return node.isEndOfWord;
    }
}

public static List<String> findClubbedWords(String[] words) {
    Trie trie = new Trie();
    for (String word : words) {
        trie.insert(word);
    }

    List<String> clubbedWords = new ArrayList<>();
    for (String word : words) {
        if (isWordFormed(word, trie, 0, 0)) {
            clubbedWords.add(word);
        }
    }

    return clubbedWords;
}

private static boolean isWordFormed(String word, Trie trie, int start, int count) {
    if (start == word.length()) {
        return count >= 2;
    }

    TrieNode node = trie.root;
    for (int i = start; i < word.length(); i++) {
        char ch = word.charAt(i);
        if (!node.children.containsKey(ch)) {
            return false;
        }
        node = node.children.get(ch);
        if (node.isEndOfWord) {
            if (isWordFormed(word, trie, i + 1, count + 1)) {
                return true;
            }
        }
    }

    return false;
}
```

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of words: ");
        int n = scanner.nextInt();
        scanner.nextLine();
        String[] words = new String[n];
        System.out.println("Enter the words:");
        for (int i = 0; i < n; i++) {
            words[i] = scanner.nextLine();
        }
        List<String> clubbedWords = findClubbedWords(words);
        System.out.println("Output: " + clubbedWords);

        scanner.close();
    }
}
```

**OUTPUT:**

```
Enter the number of words: 8
Enter the words:mat
mate
matbellmates
bell
bellmatesbell
butterribbon
butter
ribbon
Output:[matbellmates, bellmatesbell, butterribbon]
```