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

```
import java.util.*;

public class Main {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter the value of n :");
        int n = scan.nextInt();

        int[] arr = new int[n];
        int[] result;

        System.out.println("Enter the array elements :");
        for (int i = 0; i < n; i++) {
            arr[i] = scan.nextInt();
        }

        result = calculateProductArray(arr);


        System.out.println("Products of array elements :");
        for (int i = 0; i < n; i++) {
            System.out.print(result[i] + " ");
        }
    }


    private static int[] calculateProductArray(int[] nums) {
        int n = nums.length;
        int[] result = new int[n];

        for (int i = 0; i < n; i++) {
            int prod = 1;
            for (int j = 0; j < n; j++) {
                if (i != j) {
                    prod = prod * nums[j];
                }
            }
            result[i] = prod;
        }

        return result;
    }
}
```

```
Enter the value of n :
4
Enter the array elements :
1 2 3 4
Products of array elements :
24 12 8 6
Process finished with exit code 0
```

2 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]]

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    static void swap(int nums[], int l, int i) {
        int temp = nums[l];
        nums[l] = nums[i];
        nums[i] = temp;
    }

    static void permutations(ArrayList<int[]> res,
                             int[] nums, int l, int h) {

        if (l == h) {
            res.add(Arrays.copyOf(nums, nums.length));
            return;
        }


        for (int i = l; i <= h; i++) {

            swap(nums, l, i);
            permutations(res, nums, l + 1, h);
            swap(nums, l, i);
        }
    }


    static ArrayList<int[]> permute(int[] nums) {

        ArrayList<int[]> res = new ArrayList<int[]>();
        int x = nums.length - 1;
        permutations(res, nums, 0, x);
        return res;
    }


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

```
        System.out.println("Enter the length of the array:");
        int n = scan.nextInt();
        int[] nums = new int[n];

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

        ArrayList<int[]> res = permute(nums);


        for (int i = 0; i < res.size(); i++) {
            int[] x = res.get(i);
            for (int j = 0; j < x.length; j++) {
                System.out.print(x[j] + " ");
            }
            System.out.println();
        }
    }
}
```

```
Enter the length of the array:
3
Enter the elements of the array:
1 4 3
1 4 3
1 3 4
4 1 3
4 3 1
3 4 1
3 1 4

Process finished with exit code 0
```

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

```
import java.util.*;

public class Main {

    public static List<String> findClubbedWords(String[] words) {
        Set<String> wordSet = new HashSet<>(Arrays.asList(words));
        List<String> clubbedWords = new ArrayList<>();

        for (String word : words) {
            wordSet.remove(word);
```

```
            if (canFormClubbedWord(word, wordSet)) {
                clubbedWords.add(word);
            }

            wordSet.add(word);
        }

        return clubbedWords;
    }

    private static boolean canFormClubbedWord(String word, Set<String> wordSet) {
        if (wordSet.isEmpty()) {
            return false;
        }

        boolean[] dp = new boolean[word.length() + 1];
        dp[0] = true;

        for (int i = 1; i <= word.length(); i++) {
            for (int j = 0; j < i; j++) {
                if (dp[j] && wordSet.contains(word.substring(j, i))) {
                    dp[i] = true;
                    break;
                }
            }
        }

        return dp[word.length()];
    }

    public static void main(String[] args) {
        String[] words = {"mat", "mate", "matbellmates", "bell", "bellmatesbell", "butterribbon", "butter",
"ribbon"};

        List<String> clubbedWords = findClubbedWords(words);

        System.out.println("Input words: " + Arrays.toString(words));
        System.out.println("Clubbed words: " + clubbedWords);
    }
}
```