**Nested Loops**

**Easy: 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]**

**CODE:**

import java.util.Scanner;

public class Product{

public static int[] productExceptSelf(int[] nums) {

int n = nums.length;

int[] leftProducts = new int[n];

int[] rightProducts = new int[n];

leftProducts[0] = 1;

for (int i = 1; i < n; i++) {

leftProducts[i] = leftProducts[i - 1] \* nums[i - 1];

}

rightProducts[n - 1] = 1;

for (int i = n - 2; i >= 0; i--) {

rightProducts[i] = rightProducts[i + 1] \* nums[i + 1];

}

int[] result = new int[n];

for (int i = 0; i < n; i++) {

result[i] = leftProducts[i] \* rightProducts[i];

}

return result;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of elements: ");

int n = scanner.nextInt();

int[] input = new int[n];

System.out.println("Enter the elements:");

for (int i = 0; i < n; i++) {

input[i] = scanner.nextInt();

}

int[] output = productExceptSelf(input);

System.out.print("Input: [");

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

System.out.print(input[i]);

if (i < input.length - 1) {

System.out.print(", ");

}

}

System.out.println("]");

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();

}

}

**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.Arrays;

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);

used[i] = false;

currentPermutation.remove(currentPermutation.size() - 1);

}

}

}

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.print("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("Input: nums = " + Arrays.toString(nums));

System.out.println("Output: " + permutations);

scanner.close();

}

}

**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.HashMap;

import java.util.HashSet;

import java.util.List;

import java.util.Scanner;

import java.util.Set;

public class concat{

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the number of words:");

int n = scanner.nextInt();

System.out.println("Enter the words:");

String[] words = new String[n];

for (int i = 0; i < n; i++) {

words[i] = scanner.next();

}

List<String> concatenatedWords = findAllConcatenatedWords(words);

System.out.println("Concatenated words: " + concatenatedWords);

}

public static List<String> findAllConcatenatedWords(String[] words) {

HashMap<String, Boolean> hm = new HashMap<>();

Set<String> hs = new HashSet<>();

for (String word : words) {

hs.add(word);

}

List<String> result = new ArrayList<>();

for (String word : words) {

if (isConcatenated(word, hs, hm)) {

result.add(word);

}

}

return result;

}

public static boolean isConcatenated(String word, Set<String> wordSet, HashMap<String, Boolean> m) {

if (m.containsKey(word)) {

return m.get(word);

}

for (int i = 1; i < word.length(); i++) {

String prefix = word.substring(0, i);

String suffix = word.substring(i);

if (wordSet.contains(prefix) && (wordSet.contains(suffix) || isConcatenated(suffix, wordSet, m))) {

m.put(word, true);

return true;

}

}

m.put(word, false);

return false;

}

}