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]

public class ProductExceptSelf {

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

int n = nums.length;

int[] result = new int[n];

// Fill the result array with prefix products

result[0] = 1;

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

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

}

// Compute the suffix product and multiply with the existing prefix products

int suffixProduct = 1;

for (int i = n - 1; i >= 0; i--) {

result[i] \*= suffixProduct;

suffixProduct \*= nums[i];

}

return result;

}

public static void main(String[] args) {

int[] nums = {1, 2, 3, 4};

int[] result = productExceptSelf(nums);

System.out.println(Arrays.toString(result)); // Output: [24, 12, 8, 6]

}

}

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.List;

public class Permutations {

public static List<List<Integer>> permute(int[] nums) {

List<List<Integer>> results = new ArrayList<>();

helper(nums, new ArrayList<>(), results);

return results;

}

private static void helper(int[] nums, List<Integer> currentPermutation, List<List<Integer>> results) {

if (currentPermutation.size() == nums.length) {

results.add(new ArrayList<>(currentPermutation));

return;

}

for (int num : nums) {

if (currentPermutation.contains(num)) {

continue; // Skip duplicates

}

currentPermutation.add(num);

helper(nums, currentPermutation, results);

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

}

}

public static void main(String[] args) {

int[] nums = {1, 4, 3};

List<List<Integer>> permutations = permute(nums);

System.out.println(permutations); // Output: [[1, 4, 3], [1, 3, 4], [4, 1, 3], [4, 3, 1], [3, 1, 4], [3, 4, 1]]

}

}

Return all the clubbed words

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

import java.util.ArrayList;

import java.util.HashSet;

import java.util.List;

import java.util.Set;

public class ClubbedWords {

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

Set<String> wordSet = new HashSet<>(List.of(words)); // Store unique words for efficient lookup

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

for (String word : words) {

int wordLength = word.length();

// Check for clubbed words from shorter to longer prefixes

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

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

String suffix = word.substring(i);

if (wordSet.contains(prefix) && wordSet.contains(suffix)) {

clubbedWords.add(word);

break; // Move on to the next word once a clubbed word is found

}

}

}

return clubbedWords;

}

public static void main(String[] args) {

String[] words = {"mat", "mate", "matbellmates", "bell", "bellmatesbell", "butterribbon", "butter", "ribbon"};

List<String> clubbedWords = findClubbedWords(words);

System.out.println(clubbedWords); // Output: ["matbellmates", "bellmatesbell", "butterribbon"]

}

}