1. **Print Anagrams Together**

Given an array of strings, return all groups of strings that are anagrams. The groups must be created in order of their appearance in the original array. Look at the sample case for clarification.

//Initial Template for Java

import java.io.\*;

import java.util.\*;

class GFG {

public static void main (String[] args) throws IOException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int t=Integer.parseInt(br.readLine().trim());

while(t > 0)

{

int n= Integer.parseInt(br.readLine().trim());

String x = br.readLine().trim();

String string\_list[] = x.split(" ",n);

Solution ob = new Solution();

List <List<String>> ans = ob.Anagrams(string\_list);

Collections.sort(ans, new Comparator<List<String>>(){

public int compare(List<String> l1, List<String> l2) {

String s1 = l1.get(0);

String s2 = l2.get(0);

return s1.compareTo(s2);

}

});

for(int i=0;i<ans.size();i++) {

for(int j=0;j<ans.get(i).size();j++){

System.out.print(ans.get(i).get(j) + " ");

}

System.out.println();

}

t--;

}

}

}

**class Solution {**

**public List<List<String>> Anagrams(String[] string\_list) {**

**// Code here**

**HashMap <String,List<String>> mp=new HashMap<>();**

**List<List<String>> ans=new ArrayList<List<String>>();**

**for(String i:string\_list){**

**char ch[]=i.toCharArray();**

**Arrays.sort(ch);**

**String s=String.valueOf(ch);**

**if(mp.containsKey(s)){**

**mp.get(s).add(i);**

**}**

**else{**

**List<String> li=new ArrayList<>();**

**li.add(i);**

**ans.add(li);**

**mp.put(s,li);**

**}**

**}**

**return ans;**

**}**

**}**