Q1) Sort Elements of an Array by Frequency

```
import java.util.*;

public class sortArrayByFrequency {

        public static void main(String[] args)
        {
      Scanner s = new Scanner(System.in);
      int testcases = s.nextInt();

      while(testcases-->0){
         int n = s.nextInt();
                  int[] array = new int[n];

for(int i = 0; i< n; i++) {
            array[i] = s.nextInt();
         }

                  Map<Integer, Integer> map = new HashMap<>();
                  List<Integer>outputArray = new ArrayList<>();

                  for (int current : array) {
                           int count = map.getOrDefault(current, 0);
                           map.put(current, count + 1);
                           outputArray.add(current);
                  }

                  SortComparator comp = new SortComparator(map);

                  Collections.sort(outputArray, comp);

                  for (Integer i :outputArray) {
```

```
                System.out.print(i + " ");
            }
System.out.println();
        }
}
}

class SortComparator implements Comparator<Integer> {
        private final Map<Integer, Integer>freqMap;

        SortComparator(Map<Integer, Integer>tFreqMap)
        {
            this.freqMap = tFreqMap;
        }
        public int compare(Integer k1, Integer k2)
        {
            int freqCompare = freqMap.get(k2).compareTo(freqMap.get(k1));

            int valueCompare = k1.compareTo(k2);

            if (freqCompare == 0)
                return valueCompare;
            else
                return freqCompare;
        }
}
```

Q2) Longest Consecutive Subsequence

```
import java.util.*;

class LongestConsecutiveSubsequence {

   static int findLongestConseqSubseq(int arr[], int n) {
Arrays.sort(arr);

      int ans = 0, count = 0;

ArrayList<Integer> v = new ArrayList<Integer>();
v.add(arr[0]);

      for (int i = 1; i< n; i++) {
         if (arr[i] != arr[i - 1])
v.add(arr[i]);
      }


      for (int i = 0; i<v.size(); i++) {
         if (i> 0 &&v.get(i) == v.get(i - 1) + 1)
            count++;
         else
            count = 1;

ans = Math.max(ans, count);
      }
      return ans;
   }

   public static void main(String[] args) {
```

```
        Scanner s = new Scanner(System.in);


        int n = s.nextInt();
        int arr[] = new int[n];

for(int i=0;i<n;i++) arr[i] = s.nextInt();

System.out.println(findLongestConseqSubseq(arr, n));
    }
}
```

Q3) Number of ways to make sum

```
import java.util.*;

class CombinationalSum {

   static ArrayList<ArrayList<Integer>>combinationSum(ArrayList<Integer>arr, int sum) {
ArrayList<ArrayList<Integer>>ans = new ArrayList<>();
ArrayList<Integer> temp = new ArrayList<>();

      Set<Integer> set = new HashSet<>(arr);
arr.clear();
arr.addAll(set);
Collections.sort(arr);

findNumbers(ans, arr, sum, 0, temp);
      return ans;
   }

   static void findNumbers(ArrayList<ArrayList<Integer>>ans,
ArrayList<Integer>arr, int sum, int index,
ArrayList<Integer> temp) {

      if (sum == 0) {

ans.add(new ArrayList<>(temp));
         return;
      }

      for (int i = index; i<arr.size(); i++) {

         if ((sum - arr.get(i)) >= 0) {
```

```
temp.add(arr.get(i));

findNumbers(ans, arr, sum - arr.get(i), i, temp);

temp.remove(arr.get(i));
            }
        }
    }

    public static void main(String[] args) {

        Scanner s = new Scanner(System.in);
        int n = s.nextInt();
ArrayList<Integer>arr = new ArrayList<>();

for(int i = 0; i< n; i++) arr.add(s.nextInt());

        int sum =  s.nextInt();

ArrayList<ArrayList<Integer>>ans = combinationSum(arr, sum);



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

System.out.print("{");
            for (int j = 0; j <ans.get(i).size(); j++) {
System.out.print(ans.get(i).get(j) + " ");
            }
System.out.print("} ");
        }
    }
}
```