**Problems:**

**1. You are given 1 string of alphabetical characters. Each character can occur 1 or more times. Put every third character into an array (3rd, 6th, 9th,12th,..........). Print the arrays and also the maximum recurring element.**

**Solution)**

**import java.util.\*;**

**public class MaxOccThirdCharArr**

**{**

**public static void main(String[] args)**

**{**

**Scanner scan=new Scanner(System.in);**

**System.out.println("Enter the String");**

**String inputString=scan.next();**

**mostFrequent(pickMultipleOfThree(inputString));**

**}**

**static String pickMultipleOfThree(String inputString)**

**{**

**char[] c=inputString.toCharArray();**

**String elements="";**

**// to store only 3rd occurance element into array**

**for(int i=2;i<c.length;i+=3)**

**{**

**System.out.print(c[i]);**

**elements+=c[i];**

**}**

**System.out.println();**

**// System.out.println(elements);**

**return elements;**

**}**

**static void mostFrequent(String str)**

**{**

**// Create array to keep the count of individual**

**// characters and initialize the array as 0**

**int count[] = new int[256];**

**// Construct character count array from the input string.**

**int len = str.length();**

**for (int i=0; i<len; i++)**

**count[str.charAt(i)]++;**

**int max = -1; // Initialize max count**

**char result = ' '; // Initialize result**

**// Traversing through the string and maintaining the count of each character**

**for (int i = 0; i < len; i++)**

**{**

**if (max < count[str.charAt(i)])**

**{**

**max = count[str.charAt(i)];**

**result = str.charAt(i);**

**}**

**}**

**System.out.println(result);**

**}**

**}**

**2. You are given a sorted array of integers. Each element in the array occur 2 times expect one element. Find the element that occurs just once in the array in just O(logN) time and O(1) extra space.**

**Solution)**

**import java.util.\*;**

**public class ElementOccursOnce**

**{**

**public static void main(String[] args)**

**{**

**Scanner scan=new Scanner(System.in);**

**System.out.println("Enter the No of Elements");**

**int n=scan.nextInt();**

**System.out.println("Enter "+n+" no of Elements into Array");**

**int[] intArr=new int[n];**

**for(int i=0;i<intArr.length;i++)**

**{**

**intArr[i]=scan.nextInt();**

**}**

**Arrays.sort(intArr);**

**printNonRepeatingElement(intArr);**

**}**

**static void printNonRepeatingElement(int[] intArr)**

**{**

**for (int i = 0; i < intArr.length; i++)**

**{**

**boolean unique = true;**

**for (int j = 0; j < intArr.length; j++)**

**{**

**if (i != j && intArr[i] == intArr[j])**

**{**

**unique = false;**

**break;**

**}**

**}**

**if (unique)**

**{**

**System.out.println("The element that occurs just once in the Array: " +intArr[i]);**

**}**

**}**

**}**

**}**

**4. You are given an array of n elements. Your goal is to make all array elements equal. You can increment any of the elements of the array by a positive integer, k which is given as input. Find the minimum number of operations needed to make all elements of the array equal. If it is not possible to make all elements equal then print -1**

**Solution)**

**import java.util.Scanner;**

**public class makeArrayElementsEqual**

**{**

**public static void main(String[] args)**

**{**

**// Reading Input**

**Scanner scan=new Scanner(System.in);**

**System.out.println("Enter the no of Elements");**

**int n=scan.nextInt();**

**System.out.println("Enter the "+n+" no of Elements");**

**int[] intArr=new int[n];**

**for(int i=0;i<intArr.length;i++)**

**{**

**intArr[i]=scan.nextInt();**

**}**

**System.out.println("Enter Any Positive Integer for Incrementing");**

**int k=scan.nextInt();**

**// Printing No of operations Required**

**System.out.println("No of Operation required To make All Array Elements Equal :"+noOfOperationToBalanceArray(intArr,k));**

**}**

**static int noOfOperationToBalanceArray(int[] intArr,int k)**

**{**

**int operations=0;**

**int large=findLarge(intArr);**

**// checking whether the difference between Larger and other elements are divisible by k**

**for(int i=0;i<intArr.length;i++)**

**{**

**int diff=large-intArr[i];**

**if(diff%k!=0)**

**{**

**return -1;**

**}**

**operations+=diff/k;**

**}**

**System.out.println(operations);**

**return operations;**

**}**

**static int findLarge(int[] intArr)**

**{**

**// To find Largest Element in Array and to make other elements same as largest!**

**int large=0;**

**for (int i = 0; i < intArr.length; i++)**

**{**

**if(intArr[i]>large)**

**large=intArr[i];**

**}**

**System.out.println("Largest is "+large);**

**return large;**

**}**

**}**

**5.Rahul recently visited Atmana’s Cafe and was highly impressed by the food. Being a food enthusiast, he decided to enquire about the ingredients of each dish. There are N dishes represented by strings S1,S2,…,SN. Each ingredient used for making dishes in Atmana’s Cafe is represented by a lowercase English letter. For each valid dish: i, the ingredients used to make dish i correspond to characters in the string Si (note that ingredients may be used multiple times). A special ingredient is aningredient which is present in each dish at least once. Chef wants to know the number of special ingredients in Atmana's Cafe. Since Chef is too busy with work, can you help him?**

**Solution)**

**import java.util.\*;**

**public class SpecialIngridientInAtmanasCafe**

**{**

**public static void main(String[] args)**

**{**

**// Reading All the Dishes**

**Scanner scan=new Scanner(System.in);**

**System.out.println("Enter No of Dishes");**

**int N=scan.nextInt();**

**System.out.println("Enter "+N+" no of Dishes");**

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

**for(int i=0;i<N;i++)**

**{**

**dishes.add(scan.next());**

**}**

**checkSpecialIngridients(dishes);**

**}**

**static void checkSpecialIngridients(List<String> dishes)**

**{**

**char[] c=findSmallestString(dishes).toCharArray();**

**int specialIngredients=0;**

**for(int i=0;i<c.length;i++)**

**{**

**String temp=""+c[i];**

**int count=0;**

**for(int j=0;j<dishes.size();j++)**

**{**

**if(dishes.get(j).contains(temp))**

**{**

**count++;**

**}**

**}**

**if(count==dishes.size())**

**{**

**specialIngredients++;**

**}**

**}**

**System.out.println("Total no of Special Ingredients "+specialIngredients);**

**}**

**static String findSmallestString(List<String> dishes)**

**{**

**String str="";**

**int len=dishes.get(0).length();**

**for(int i=0;i<dishes.size();i++)**

**{**

**if(len>dishes.get(i).length())**

**{**

**len=dishes.get(i).length();**

**str=dishes.get(i);**

**}**

**}**

**return str;**

**}**

**}**

**6. Given a one dimensional integer array. Find a contiguous subarray within it that has the largest sum.**

**Solution)**

**import java.util.\*;**

**public class SubarrayWithLargestSum**

**{**

**public static void main(String[] args)**

**{**

**Scanner scan=new Scanner(System.in);**

**System.out.println("Enter the no of Whole Numbers");**

**int N=scan.nextInt();**

**System.out.println("Enter "+N+" no of Whole Numbers");**

**int[] a=new int[N];**

**for (int i = 0; i < a.length; i++)**

**{**

**a[i]=scan.nextInt();**

**}**

**getSubArrayWithLargeSum(a);**

**}**

**static void getSubArrayWithLargeSum(int[] a)**

**{**

**int max\_so\_far = Integer.MIN\_VALUE,max\_ending\_here = 0, start = 0,**

**end = 0, s = 0, size=a.length;**

**// getting subarray which has largest sum**

**for (int i = 0; i < size; i++)**

**{**

**max\_ending\_here += a[i];**

**if (max\_so\_far < max\_ending\_here)**

**{**

**max\_so\_far = max\_ending\_here;**

**start = s;**

**end = i;**

**}**

**if (max\_ending\_here < 0)**

**{**

**max\_ending\_here = 0;**

**s = i + 1;**

**}**

**}**

**System.out.print("Subarray with largest sum is { ");**

**for(int i=start;i<=end;i++)**

**{**

**System.out.print(a[i]+" ");**

**}**

**System.out.print("} with sum "+ max\_so\_far);**

**}**

**}**