**MCCP – PROBLEMS**

ROLL NUMBER: **21A81A05B4**

NAME: **N. CHANDU**

SECTION:  **CSE-B**

**Q1.**

**A) Find GCD of most occurring and least occurring elements of given Array**

Given an [array](https://www.geeksforgeeks.org/array-data-structure/) **arr[]** of size **n**, The task is to find the [GCD](http://www.geeksforgeeks.org/basic-and-extended-euclidean-algorithms/) of the [highest and lowest frequency element in the given array.](https://www.geeksforgeeks.org/difference-between-highest-and-least-frequencies-in-an-array/)

**Examples:**

***Input:*** *arr[] = {2, 2, 4, 4, 5, 5, 6, 6, 6, 6}*

***Output:*** *2*

***Explanation:*** *The frequency of the elements in the above array is freq(2) = 2, freq(4) = 2, freq(5) = 2, freq(6) = 4,*

*The minimum frequency is 2 (of elements 2, 4, and 5). So 2 will be picked as the least among 2, 4, and 5.*

*The largest frequency is 4 (of element 6). Hence GCD of 2 and 6 = gcd(2, 6) is 2.*

***Input:*** *arr[] = {3, 2, 2, 44, 44, 44, 44}*

***Output:*** *1*

**Code**

import java.util.HashMap; import java.util.Map; import java.util.Scanner;

public class GCDOfFrequencyElements { public static void main(String[] args) { Scanner scanner = new Scanner(System.in); int n = scanner.nextInt(); int[] arr = new int[n];

System.out.println("Enter the elements of the array:"); for (int i = 0; i < n; i++) { arr[i] = scanner.nextInt();

}

int gcd = findGCDOfFrequencyElements(arr);

System.out.println(gcd);

}

private static int findGCDOfFrequencyElements(int[] arr) { Map<Integer, Integer> frequencyMap = new HashMap<>(); for (int num : arr) {

frequencyMap.put(num, frequencyMap.getOrDefault(num, 0) + 1);

}

int maxFrequencyElement = arr[0]; for (int num : frequencyMap.keySet()) {

if (frequencyMap.get(num) > frequencyMap.get(maxFrequencyElement)) { maxFrequencyElement = num;

}

}

int minFrequencyElement = arr[0]; for (int num : frequencyMap.keySet()) {

if (frequencyMap.get(num) < frequencyMap.get(minFrequencyElement)) { minFrequencyElement = num;

}

}

return findGCD(maxFrequencyElement, minFrequencyElement);

}

private static int findGCD(int a, int b) { if (b == 0) { return a;

}

return findGCD(b, a % b);

}

}

**B) Develop Program to find the initials of a name.** Given a string name, we have to find the initials of the name

**Examples:**

Input : prabhat kumar singh

Output : P K S

We take the first letter of all words and print in capital letter.

Input : Jude Law

Output : J L

Input : abhishek kumar singh

Output : A K S

**Code :**

import java.util.Scanner; public class InitialsFind{

public static void main(String[] args) { Scanner scanner = new Scanner(System.in); String name = scanner.nextLine(); findAndPrintInitials(name);

}

private static void findAndPrintInitials(String name) { String[] words = name.split(" "); for (String word : words) { if (!word.isEmpty()) {

System.out.print(Character.toUpperCase(word.charAt(0)) + " ");

}

}

}

}

Q2)

**A) Check Whether a number is Duck Number or not**

A Duck number is a positive number which has zeroes present in it, For example 3210, 8050896, 70709 are all Duck numbers. Please note that a numbers with only leading 0s is not considered as Duck Number. For example, numbers like 035 or 0012 are not considered as Duck Numbers. A number like 01203 is considered as Duck because there is a non-leading 0 present in it.

**Examples :**

*Input : 707069*

*Output : It is a duck number.*

*Explanation: 707069 does not contains zeros at the beginning.*

*Input : 02364*

*Output : It is not a duck number.*

*Explanation: in 02364 there is a zero at the beginning of the number.*

Time Complexity:O(n) **where n is length of string.**

Auxiliary Space: O(1)

**Code**

import java.util.Scanner; public class DuckNumberChecker { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); String numberStr = scanner.nextLine(); if (isDuckNumber(numberStr)) {

System.out.println("It is a duck number.");

} else {

System.out.println("It is not a duck number.");

}

}

private static boolean isDuckNumber(String numberStr) { if (numberStr.charAt(0) == '0') {

return false;

}

for (int i = 1; i < numberStr.length(); i++) { if (numberStr.charAt(i) == '0') { return true;

} } return false;

}

}

**B) Find Kth most occurring element in an Array**

Given an array of integers **arr[]** of size **N** and a number **K**, the task is to find the **Kth** most occurring element in this array.

**Note:** If there are more than one numbers in the array with the same frequency, then both are considered to be at the same level of occurrence. Therefore print both the numbers.

**Examples:**

***Input:*** *arr[] = {1, 2, 2, 2, 4, 4, 4, 5, 5, 5, 5, 5, 7, 7, 8, 8, 8, 8}, K = 1*

***Output:*** *5*

***Explanation:***

*The occurrence of the elements are as follows:*

1. *– 1*
2. *– 3*
3. *– 3*
4. *– 5*
5. *– 2*
6. *– 4*

*Clearly, 5 is the most occurring element in the array.*

***Input:*** *arr[] = {1, 2, 2, 2, 4, 4, 4, 4, 5, 5, 5, 5, 5, 7, 7, 8, 8, 8, 8}, K = 3*

***Output:*** *2*

**Code**

import java.util.HashMap; import java.util.Map; import java.util.PriorityQueue; import java.util.Scanner; public class KthMostOccurringElement {

public static void main(String[] args) { Scanner scanner = new Scanner(System.in);

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

}

int k = scanner.nextInt();

findAndPrintKthMostOccurringElement(arr, k);

}

private static void findAndPrintKthMostOccurringElement(int[] arr, int k) { Map<Integer, Integer> frequencyMap = new HashMap<>();

for (int num : arr) {

frequencyMap.put(num, frequencyMap.getOrDefault(num, 0) + 1);

}

// Create a max heap (PriorityQueue) based on frequency

PriorityQueue<Map.Entry<Integer, Integer>> maxHeap = new PriorityQueue<>( (a, b) -> b.getValue().compareTo(a.getValue())

);

maxHeap.addAll(frequencyMap.entrySet()); System.out.println("Kth most occurring element(s):");

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

Map.Entry<Integer, Integer> entry = maxHeap.poll();

System.out.println(entry.getKey());

}

}

}

Q3.

**A) Removing punctuations from a given string**

Given a string, remove the punctuation from the string if the given character is a punctuation character, as classified by the current C locale. The default C locale classifies these characters as punctuation:

**! " # $ % & ' ( ) \* + , - . / : ; ? @ [ \ ] ^ \_ ` { | } ~**  **Examples:**

**Input :** %welcome' to @geeksforgeek<s

**Output :** welcome to geeksforgeeks

**Input :** Hello!!!, he said ---and went.

**Output :** Hello he said and went

Code

import java.util.Scanner; public class RemovePunctuation {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String inputString = scanner.nextLine();

String result = removePunctuation(inputString);

System.out.println( result);

}

private static String removePunctuation(String inputString) { StringBuilder result = new StringBuilder(); for (char ch : inputString.toCharArray()) { if (!isPunctuation(ch)) { result.append(ch);

}

}

return result.toString();

}

private static boolean isPunctuation(char ch) { return !(ch >= 'A' && ch <= 'Z' || ch >= 'a' && ch <= 'z');

}

}

**B) Print all array elements occurring at least M times**

Given an [array](https://www.geeksforgeeks.org/introduction-to-arrays/) **arr[]** consisting of **N** integers and a positive integer **M**, the task is to find the number of array elements that occur **at least M** times.

**Examples:**

***Input:*** *arr[] = {2, 3, 2, 2, 3, 5, 6, 3}, M = 2* ***Output:*** *2 3*

***Explanation:***

*In the given array arr[], the element that occurs at least M number of times are {2, 3}.* ***Input:*** *arr[] = {1, 32, 2, 1, 33, 5, 1, 5}, M = 2*

***Output:*** *1 5*

**Code**

import java.util.HashMap; import java.util.Map; import java.util.Scanner; public class ElementsAtLeastMTimes { public static void main(String[] args) { Scanner scanner = new Scanner(System.in); int n = scanner.nextInt(); int[] arr = new int[n]; for (int i = 0; i < n; i++) { arr[i] = scanner.nextInt();

}

int m = scanner.nextInt(); printElementsAtLeastMTimes(arr, m); }

private static void printElementsAtLeastMTimes(int[] arr, int m) { Map<Integer, Integer> frequencyMap = new HashMap<>(); for (int num : arr) {

frequencyMap.put(num, frequencyMap.getOrDefault(num, 0) + 1);

}

System.out.println("Array elements occurring at least " + m + " times:"); for (Map.Entry<Integer, Integer> entry : frequencyMap.entrySet()) { if (entry.getValue() >= m) {

System.out.print(entry.getKey() + " ");

}

}

} }

**Q4) A) Check whether the given number is Euclid Number or not**

Given a positive integer n, the task is to check if it is Euclid Number or not. Print ‘YES’ if the given number is Euclid Number otherwise print ‘NO’.

[**Euclid number :**](https://en.wikipedia.org/wiki/Euclid_number) In Mathematics, Euclid numbers are integers of the form – E=P#+1

where P# is product of first n prime numbers. The first few Euclid numbers are-

*3, 7, 31, 211, 2311, 30031, 510511, 9699691, ……….*

**Input:** N = 31

**Output:** YES

31 can be expressed in the form of pn# + 1 as p3# + 1

(First 3 prime numbers are 2, 3, 5 and their product is 30 )

**Input:** N = 43

**Output:** NO

43 Cannot be expressed in the form of pn# + 1

import java.math.BigInteger; import java.util.Scanner; public class EuclidNumberChecker { public static boolean isEuclidNumber(int n) { BigInteger p = BigInteger.ONE;

for (int i = 1; i <= n; i++) { p = p.multiply(BigInteger.valueOf(i));

}

BigInteger euclidNumber = p.add(BigInteger.ONE); return euclidNumber.isProbablePrime(10);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); System.out.print("Enter a positive integer: "); int n = scanner.nextInt(); if (isEuclidNumber(n)) {

System.out.println(n + " is a Euclid Number");

} else {

System.out.println(n + " is not a Euclid Number");

}

}

}

**B) Program to print reverse character bridge pattern**

For a given value N, denoting the number of Charters starting from the A, print reverse character bridge pattern. **Examples :** Input : n = 5 Output :

ABCDEDCBA

ABCD DCBA

ABC CBA

AB BA

A A

Input : n = 8

Output :

ABCDEFGHGFEDCBA

ABCDEFG GFEDCBA

ABCDEF FEDCBA

ABCDE EDCBA

ABCD DCBA

ABC CBA

AB BA

A A

public class ReverseCharBridge { public static void main(String[] args) { Scanner sc=new Scanner(System.in);

int n=sc.nextInt(); for(int i=0;i<n;i++)

{ if(i==0)

for(int j=0;j<n-i-1;j++) System.out.print((char)('A'+j));

else {

for(int j=0;j<n-i;j++)

System.out.print((char)('A'+j)); }

for(int l=0;l<i;l++) System.out.print(" "); for(int k=n-i-1;k>=0;k--) System.out.print((char)('A'+k));

System.out.println();

}

} }

Q5)

**A. Sum of all Perfect numbers lying in the range [L, R]**

Given two numbers **L**, **R** which signifies the range **[L, R]**, the task is to find the sum of all [perfect numbers](https://www.geeksforgeeks.org/perfect-number/) lying in the range [L, R].

**Examples:**

***Input:*** *L = 6, R = 10* ***Output:*** *6*

***Explanation:***

*From 6 to 10, the only perfect number is 6.*

***Input:*** *L = 6, R = 28* ***Output:*** *34*

***Explanation:***

*There are two perfect numbers in the range [6, 28]. They are, {6, 28} 6 + 28 = 34.*

import java.util.Scanner; public class PerfectNumberSum { public static boolean isPerfectNumber(int num) {

int sum = 0; for (int i = 1; i < num; i++) { if (num % i == 0) { sum += i;

}

}

return sum == num;

}

public static int getPerfectNumberSum(int L, int R) {

int sum = 0; for (int i = L; i <= R; i++) { if (isPerfectNumber(i)) { sum += i;

}

}

return sum;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); System.out.print("Enter the value of L: "); int L = scanner.nextInt();

System.out.print("Enter the value of R: "); int R = scanner.nextInt();

int perfectNumberSum = getPerfectNumberSum(L, R);

System.out.println("Sum of perfect numbers in the range [" + L + ", " + R + "]: " +

perfectNumberSum);

}

}

**B) Count words in a given string**

Given a string, count the number of words in it. The words are separated by the following characters: space (‘ ‘) or new line (‘\n’) or tab (‘\t’) or a combination of these.

**Input:** S = "abc def"

**Output:** 2

**Input:** S = "a\nyo\n" **Output:** 2

public class WordsCount { public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

String str=sc.nextLine();

int i=0; int f=1; int c=0; while(i<str.length())

{

if(str.charAt(i) == ' ' || str.charAt(i) == '\n'

|| str.charAt(i) == '\t')

f=1; else if(f==1) { c++; f=0; } i++;

}

System.out.println(c);

} }

Q6)

**A) Find one extra character in a string**

Given two strings which are of lengths n and n+1. The second string contains all the character of the first string, but there is one extra character. Your task to find the extra character in the second string.

**Examples :**

**Input :** string strA = "abcd";

string strB = "cbdae"; **Output :** e

string B contain all the element there is a one extra character which is e **Input :** string strA = "kxml"; string strB = "klxml"; **Output :** l string B contain all the element

there is a one extra character which is l

import java.util.HashMap; import java.util.Map;

import java.util.Scanner;

public class ExtraCharacterFinder { public static char findExtraCharacter(String str1, String str2) {

Map<Character, Integer> charCount = new HashMap<>();

// Count characters in str1 for (char ch : str1.toCharArray()) {

charCount.put(ch, charCount.getOrDefault(ch, 0) + 1);

}

// Subtract characters in str2 for (char ch : str2.toCharArray()) {

charCount.put(ch, charCount.getOrDefault(ch, 0) - 1);

}

// Find the extra character

for (Map.Entry<Character, Integer> entry : charCount.entrySet()) { if (entry.getValue() != 0) {

return entry.getKey();

}

}

return '\0'; // No extra character found

}

public static void main(String[] args) { Scanner scanner = new Scanner(System.in);

System.out.print("Enter the first string: ");

String str1 = scanner.nextLine();

System.out.print("Enter the second string: ");

String str2 = scanner.nextLine();

char extraCharacter = findExtraCharacter(str1, str2);

if (extraCharacter != '\0') {

System.out.println("Extra character: " + extraCharacter);

} else {

System.out.println("No extra character found");

}

}

}

B) Given an array A with N integers, find the count of unique integers in the array.

Input : arr = [3 2 4 1 2 3]

Output : 2

Explanation: only 1 and 4 are unique in the array rest 2, 3 are

occurring 2 times.

**Time Complexity: O(N)**

**Space Complexity**

:

O(N)

public class CountofUnique {

public static void main(String[] args) { Scanner sc=new Scanner(System.in); int n=sc.nextInt(); int a[]=new int[n]; for(int i=0;i<n;i++) a[i]=sc.nextInt();

HashMap<Integer,Integer>hm=new HashMap<>();

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

hm.put(a[i],hm.getOrDefault(a[i],0)+1); int c=0;

for(int i: hm.keySet()) if(hm.get(i)==1) c++;

System.out.print(c);

} }

Q7)

**A) Program to print the initials of a name with the surname**

Given a full name in the form of a string, the task is to print the initials of a name, in short, and surname in full.

**Examples:**

**Input:** Devashish Kumar Gupta

**Output:** D. K. Gupta

**Input:** Ishita Bhuiya

**Output:** I. Bhuiya

import java.util.Scanner; public class InitialsPrinter { public static void printInitials(String fullName) {

String[] nameParts = fullName.trim().split("\\s+"); StringBuilder initials = new StringBuilder(); for (String namePart : nameParts) { initials.append(namePart.charAt(0));

}

System.out.println("Initials: " + initials.toString().toUpperCase());

System.out.println("Surname: " + nameParts[nameParts.length - 1]);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the full name: "); String fullName = scanner.nextLine(); printInitials(fullName);

}

}

B) Given an array of size N, which contains the voting ID's of students that have stood up for the elections for class monitor, the candidate with votes greater than half the strength of the class will become monitor, find the ID of candidate that can become monitor else return -1 if no one can become.

|  |  |  |
| --- | --- | --- |
| Input : A = [1, 3, 2, 2, 2] Output : 2  Explanation : 2 got 3 votes which is greater than half the strength of the class i.e. 5/2 = 2. | | |
|  | | |
| **Time Complexity: O(N)** |  | |
|  |
| **Space Complexity** : O(N) | |  |
|  | |

public class VoterIds { public static void main(String[] args) { Scanner sc= new Scanner(System.in);

int n=sc.nextInt(); int a[]=new int[n]; for(int i=0;i<n;i++) a[i]=sc.nextInt(); for(int i=0;i<n;i++) if(a[i]>n/2)

System.out.print(i+1);

}

}

Q8)

**A) Check if frequency of character in one string is a factor or multiple of frequency of same character in other string**

Given two strings, the task is to check whether the frequencies of a character(for each character) in one string are multiple or a factor in another string. If it is, then output “YES”, otherwise output “NO”.

**Examples:**

***Input:*** *s1 = “aabccd”, s2 = “bbbaaaacc”*

***Output:*** *YES*

*Frequency of ‘a’ in s1 and s2 are 2 and 4 respectively, and 2 is a factor of 4 Frequency of ‘b’ in s1 and s2 are 1 and 3 respectively, and 1 is a factor of 3 Frequency of ‘c’ in s1 and s2 are same hence it also satisfies.*

*Frequency of ‘d’ in s1 and s2 are 1 and 0 respectively, but 0 is a multiple of every number, hence satisfied.*

*Hence, the answer YES.*

***Input:*** *s1 = “hhdwjwqq”, s2 = “qwjdddhhh”*

***Output:*** *NO*

import java.util.HashMap;

import java.util.Map; import java.util.Scanner; public class FrequencyCheck { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the first string: ");

String str1 = scanner.nextLine();

System.out.print("Enter the second string: "); String str2 = scanner.nextLine(); boolean result = checkFrequency(str1, str2);

if (result) {

System.out.println("YES");

} else {

System.out.println("NO");

} scanner.close();

}

public static boolean checkFrequency(String str1, String str2) { if (str1.length() != str2.length()) { return false;

}

Map<Character, Integer> freqMap1 = new HashMap<>(); Map<Character, Integer> freqMap2 = new HashMap<>(); for (char c : str1.toCharArray()) { freqMap1.put(c, freqMap1.getOrDefault(c, 0) + 1);

}

for (char c : str2.toCharArray()) { freqMap2.put(c, freqMap2.getOrDefault(c, 0) + 1);

}

for (char c : freqMap1.keySet()) { if (freqMap2.containsKey(c)) { int freq1 = freqMap1.get(c); int freq2 = freqMap2.get(c); if (freq1 % freq2 != 0 && freq2 % freq1 != 0) {

return false;

} } else { return false;

} } return true;

}

}

B) Given marks of N students sitting on a bench and a value of K, print the index of the student whose marks matches with the value of K.

Input : N = 10, K = 67

A[] = [60, 61, 62, 63, 63, 64, 65, 66, 67, 66]

Output : 8

Explanation : 67 is present at 8th index (0-based indexing)

import java.util.Scanner; public class StudentMarks { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of students (N): "); int N = scanner.nextInt();

System.out.print("Enter the value to match (K): "); int K = scanner.nextInt(); int[] marks = new int[N];

System.out.println("Enter the marks of the students:");

for (int i = 0; i < N; i++) { marks[i] = scanner.nextInt();

}

int index = -1; for (int i = 0; i < marks.length; i++) { if (marks[i] == K) {

index = i; break;

} }

if (index != -1) {

System.out.println("Index of the student with marks " + K + " is " + index); } else {

System.out.println("No student found with marks " + K);

}

scanner.close();

}

}

Q9)

**A) Check if given String is Pangram or not**

Given a string **Str**.The task is to check if it is Pangram or not.

*A* ***pangram*** *is a sentence containing every letter in the English Alphabet.* **Examples:**

***Input:*** *“The quick brown fox jumps over the lazy dog”* ***Output:*** *is a Pangram*

***Explanation:*** *Contains all the characters from ‘a’ to ‘z’]*

import java.util.Scanner; public class PangramChecker { public static boolean isPangram(String str) { boolean[] alphabet = new boolean[26]; str = str.toLowerCase(); for (int i = 0; i < str.length(); i++) { char c = str.charAt(i); if (Character.isLetter(c)) { alphabet[c - 'a'] = true;

} present for (boolean present : alphabet) { if (!present) { return false;

} } return true; }

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: "); String str = scanner.nextLine(); scanner.close(); if (isPangram(str)) {

System.out.println("The string is a pangram");

} else {

System.out.println("The string is not a pangram");

}

}

}

B) The Leaders

Print all those elements that have no element greater than them in the right side of the array. Print elements from right to left.

Test Case 1:

Input : A[] = [1, 2, 3, 4, 5]

Output : 5

Explanation : Only 5 has no element greater than it on its right.

Test Case 2:

Input : A[] = [1, 4, 3, 2]

Output : 2 3 4

Explanation : 2, 3, and 4 have no elements greater than them on their right while 1 has 4, 3 and 2 greater than it on its right.

import java.util.Scanner; public class LeadersInArray {

public static void printLeaders(int[] arr) {

int n = arr.length; int maxSoFar = arr[n - 1]; System.out.print(maxSoFar + " "); for (int i = n - 2; i >= 0; i--) { if (arr[i] > maxSoFar) { maxSoFar = arr[i];

System.out.print(maxSoFar + " ");

}

}

}

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[] arr = new int[n];

System.out.println("Enter the elements of the array: "); for (int i = 0; i < n; i++) {

arr[i] = scanner.nextInt();

}

scanner.close();

System.out.print("Leaders in the array: "); printLeaders(arr);

}

}

Q10)

**A) Missing characters to make a string Pangram**

[Pangram](https://www.geeksforgeeks.org/pangram-checking/) is a sentence containing every letter in the English alphabet. Given a string, find all characters that are missing from the string, i.e., the characters that can make the string a Pangram. We need to print output in alphabetic order.

**Examples:**

Input : welcome to geeksforgeeks

Output : abdhijnpquvxyz

Input : The quick brown fox jumps

Output : adglvyz

import java.util.HashSet; import java.util.Scanner; public class MissingCharacters { public static String getMissingCharacters(String str) { HashSet<Character> uniqueChars = new HashSet<>();

str = str.toLowerCase(); for (int i = 0; i < str.length(); i++) { char c = str.charAt(i); if (Character.isLetter(c)) { uniqueChars.add(c);

}

} alphabetic order

StringBuilder missingChars = new StringBuilder();

for (char c = 'a'; c <= 'z'; c++) { if (!uniqueChars.contains(c)) { missingChars.append(c);

}

}

return missingChars.toString();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: "); String str = scanner.nextLine(); scanner.close();

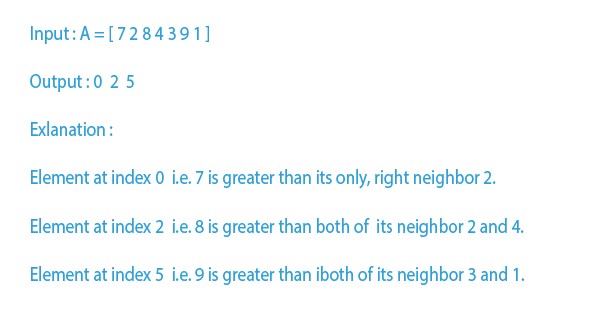
String missingChars = getMissingCharacters(str);

System.out.println("Missing characters to make the string a Pangram: " + missingChars);

}

}

B) Given an array A of N elements, your task is to print all those indexes that have values greater than its left and right neighbors. In case of extreme indexes like 0 and N-1, check with their single neighbor.



import

java.util.Scanner; public class GreaterThanNeighbors {

public static void printIndexes(int[] arr) { int n = arr.length; if (n > 1 && arr[0] > arr[1]) {

System.out.println("Index 0");

}

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

if (arr[i] > arr[i - 1] && arr[i] > arr[i + 1]) {

System.out.println("Index " + i);

}

}

if (n > 1 && arr[n - 1] > arr[n - 2]) {

System.out.println("Index " + (n - 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[] arr = new int[n];

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

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

}

scanner.close();

System.out.println("Indexes with values greater than neighbors:"); printIndexes(arr);

}

}

Q11)

**A) Check if max occurring character of one string appears same no. of times in other**

Given two strings, we need to take the character which has the maximum occurrence in the first string, and then we have to check if that particular character is present in the second string the same number of times as it is present in the first string.

**Examples:**

Input : s1 = "sssgeek", s2 = "geeksss"

Output : Yes

Max occurring character in s1 is 's'. It occurs same number of times in s2.

Input : geekyarticle gfggfggfg

Output : No

import java.util.HashMap; import java.util.Map; import java.util.Scanner; public class MaxOccurringCharacter {

public static boolean checkMaxOccurringCharacter(String s1, String s2) { Map<Character, Integer> frequencyMap = new HashMap<>(); for (char c : s1.toCharArray()) { frequencyMap.put(c, frequencyMap.getOrDefault(c, 0) + 1);

}

char maxChar = '\0'; int maxCount = 0;

for (Map.Entry<Character, Integer> entry : frequencyMap.entrySet()) { if (entry.getValue() > maxCount) { maxChar = entry.getKey(); maxCount = entry.getValue();

} } int count = 0; for (char c : s2.toCharArray()) { if (c == maxChar) { count++;

}

}

return count == maxCount;

}

public static void main(String[] args) { Scanner scanner = new Scanner(System.in);

System.out.print("Enter the first string: ");

String s1 = scanner.nextLine();

System.out.print("Enter the second string: "); String s2 = scanner.nextLine();

boolean result = checkMaxOccurringCharacter(s1, s2); System.out.println(result ? "Yes" : "No"); scanner.close();

} }

|  |  |
| --- | --- |
| **B)** | **Cumulative Sum** |

The cumulative sum of an array at index i is defined as the sum of all elements of the array from index 0 to index i.

Examples

|  |
| --- |
| Initial Array: [1, 2, 3, 4] |
| Cumulative Sum: [1, 3, 6, 10] |
| Initial Array: [1, 1, 1, 1, 1]  Cumulative Sum: [1, 2, 3, 4, 5] |
| Initial Array: [1, 3, 5, 7, 9]  Cumulative Sum: [1, 4, 9, 16, 25] |

Given an array, return its cumulative sum.

import java.util.Scanner; public class CumulativeSum {

public static int[] calculateCumulativeSum(int[] arr) { int n = arr.length;

int[] cumulativeSum = new int[n]; cumulativeSum[0] = arr[0]; for (int i = 1; i < n; i++) {

cumulativeSum[i] = cumulativeSum[i - 1] + arr[i];

}

return cumulativeSum;

}

public static void main(String[] args) { Scanner scanner = new Scanner(System.in); System.out.print("Enter the size of the array: "); int size = scanner.nextInt(); int[] arr = new int[size];

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

for (int i = 0; i < size; i++) { arr[i] = scanner.nextInt();

}

int[] cumulativeSum = calculateCumulativeSum(arr);

System.out.print("Initial Array: ["); for (int i = 0; i < arr.length; i++) { System.out.print(arr[i]); if (i != arr.length - 1) { System.out.print(", ");

}

}

System.out.println("]");

System.out.print("Cumulative Sum: ["); for (int i = 0; i < cumulativeSum.length; i++) { System.out.print(cumulativeSum[i]); if (i != cumulativeSum.length - 1) {

System.out.print(", ");

}

}

System.out.println("]");

scanner.close();

}

}

Q12) A) Given an array of words and a string, we need to count all words that are present in given string.

**Examples:** Input : words[] = { "welcome", "to", "geeks", "portal"} str = "geeksforgeeks is a computer science portal for geeks." **Output :** 2

Two words "portal" and "geeks" is present in str.

Input : words[] = {"Save", "Water", "Save", "Yourself"} str = "Save"

**Output :**1

import java.util.Scanner; public class WordCount {

public static void main(String[] args) { Scanner scanner = new Scanner(System.in); System.out.print("Enter the number of words: "); int numWords = scanner.nextInt();

scanner.nextLine();

String[] words = new String[numWords]; System.out.println("Enter the words:"); for (int i = 0; i < numWords; i++) {

words[i] = scanner.nextLine();

}

System.out.print("Enter the string: "); String str = scanner.nextLine(); int count = 0; for (String word : words) { if (str.contains(word)) {

count++;

} }

System.out.println("Number of words present in the string: " + count);

}

}

**B) Identical Twins**

For an array of integers

nums

, an identical twin is defined as pair

)

(

i, j

where

nums[i

]

is

equal

to

nums[j]

and

i < j

.

|  |
| --- |
| Test Case 1: Array: [1, 2, 3, 2, 1] |
| Number of Identical Twins: 2 Explanation:  Identical Twins: [[1, 1], [2, 2]]  Indexes: (0, 4), (1, 3)  Test Case 2: Array: [1, 2, 2, 3, 2, 1]  Number of Identical Twins: 4 |
| Explanation: |
| Identical Twins: [[1, 1], [2, 2], [2, 2], [2, 2]] |
| Indexes: (0, 5), (1, 2), (1, 4), (2, 4) |
| Test Case 3: Array: [1, 1, 1, 1]  Number of Identical Twins: 6 Explanation:  Identical Twins: [[1, 1], [1, 1], [1, 1], [1, 1], [1, 1], [1, 1]]  Indexes: (0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3) |

Given an array nums, find the number of identical twins.

import java.util.ArrayList; import java.util.HashMap;

import java.util.List; import java.util.Map; import java.util.Scanner; public class IdenticalTwins { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of elements in the array: "); int numElements = scanner.nextInt(); int[] nums = new int[numElements];

System.out.println("Enter the elements of the array:"); for (int i = 0; i < numElements; i++) { nums[i] = scanner.nextInt();

}

int count = 0;

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

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

if (map.containsKey(nums[i])) {

List<Integer> indexes = map.get(nums[i]);

for (int index : indexes) {

System.out.println("Identical Twins: [" + nums[i] + ", " + nums[index] + "]");

System.out.println("Indexes: (" + index + ", " + i + ")");

count++;

}

indexes.add(i);

} else {

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

indexes.add(i); map.put(nums[i], indexes);

}

}

System.out.println("Number of Identical Twins: " + count);

}

}

Q13)

**A) Count words that appear exactly two times in an array of words**

Given an array of n words. Some words are repeated twice, we need to count such words.

**Examples:**

**Input :** s[] = {"hate", "love", "peace", "love",

"peace", "hate", "love", "peace",

"love", "peace"};

**Output :** 1

There is only one word "hate" that appears twice

**Input :** s[] = {"Om", "Om", "Shankar", "Tripathi",

"Tom", "Jerry", "Jerry"};

**Output :** 2

There are two words "Om" and "Jerry" that appear

twice.

import java.util.HashMap; import java.util.Map; import java.util.Scanner; public class WordCounter {

public static int countWords(String[] words) {

Map<String, Integer> wordCount = new HashMap<>(); for (String word : words) {

wordCount.put(word, wordCount.getOrDefault(word, 0) + 1);

}

int count = 0;

for (Map.Entry<String, Integer> entry : wordCount.entrySet()) { if (entry.getValue() == 2) {

count++;

}

}

return count;

}

public static void main(String[] args) { Scanner scanner = new Scanner(System.in); System.out.print("Enter the number of words: ");

int n = scanner.nextInt(); String[] words = new String[n];

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

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

words[i] = scanner.next();

}

int count = countWords(words);

System.out.println("Count of words appearing exactly two times: " + count); scanner.close();

} }

|  |  |
| --- | --- |
| **B)** | **Even Number of Digits** |

Given an array of integers, find the elements which have an even number of digits.

# Example

|  |
| --- |
| Array: [42, 564, 5775, 34, 123, 454, 1, 5, 45, 3556, 23442] |
| Answer: 42, 5775, 34, 45, 3556 |

The order of the returned elements should be the same as the order of the initial array.

import java.util.ArrayList; import java.util.List; import java.util.Random; public class EvenNumberOfDigits {

public static void main(String[] args) { int[] array = generateRandomArray(10, 1, 10000);

List<Integer> result = findEvenNumberOfDigits(array);

System.out.println("Answer: " + result);

}

public static List<Integer> findEvenNumberOfDigits(int[] array) { List<Integer> result = new ArrayList<>(); for (int num : array) { int digitCount = countDigits(num); if (digitCount % 2 == 0) { result.add(num);

} }

return result;

}

public static int countDigits(int num) { if (num == 0) {

return 1;

}

int count = 0; while (num != 0) { num /= 10; count++;

}

return count;

}

public static int[] generateRandomArray(int size, int min, int max) { int[] array = new int[size];

Random rand = new Random();

for (int i = 0; i < size; i++) { array[i] = rand.nextInt(max - min + 1) + min;

}

return array;

}

}

Q14)

**A) Count of camel case characters present in a given string**

Given a [string](https://www.geeksforgeeks.org/string-data-structure/) **S**, the task is to count the number of [camel case characters](https://www.geeksforgeeks.org/camel-case-given-sentence/) present in the given string.

*The camel case character is defined as the number of uppercase characters in the given string.*

**Examples:**

***Input:*** *S = “ckjkUUYII”* ***Output:*** *5*

***Explanation:***

*Camel case characters present are U, U, Y, I and I.*

***Input:*** *S = “abcd”*

***Output:*** *0*

import java.util.Scanner; public class CamelCaseCount { public static int countCamelCaseCharacters(String str) {

int count = 0; for (int i = 0; i < str.length(); i++) { char c = str.charAt(i); if (Character.isUpperCase(c)) {

count++;

}

}

return count;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: "); String inputString = scanner.nextLine(); int count = countCamelCaseCharacters(inputString);

System.out.println("Count of camel case characters: " + count);

}

}

|  |  |
| --- | --- |
| **B)** | **Max Consecutive Ones** |

Given an array A, find the maximum number of consecutive 1s in the array.

***Examples***

A: [1, 1, 3, 2, 3, 1, 1, 1]

Max consecutive 1s: 3

|  |
| --- |
| import java.util.Scanner; public class MaxConsecutiveOnes { public static int findMaxConsecutiveOnes(int[] nums) { int maxConsecutiveOnes = 0; int currentConsecutiveOnes = 0; for (int num : nums) { if (num == 1) { currentConsecutiveOnes++;  maxConsecutiveOnes = Math.max(maxConsecutiveOnes, currentConsecutiveOnes);  } else { currentConsecutiveOnes = 0;  }  }  return maxConsecutiveOnes;  }  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.println("Enter the elements of the array:"); for (int i = 0; i < n; i++) { nums[i] = scanner.nextInt();  }  int maxConsecutiveOnes = findMaxConsecutiveOnes(nums); |

System.out.println("Max consecutive 1s: " + maxConsecutiveOnes);

}

}

**Q15)**

**A)Find resultant string after concatenating uncommon characters of given strings**

Given two strings **S1** and **S2.** The task is to concatenate uncommon characters of the **S2** to **S1** and return the resultant string **S1 .** **Examples:**

***Input:******S1*** *= “aacdb”,* ***S2*** *= “gafd”*

***Output:*** *“cbgf”*

***Input:******S1*** *= “abcs”,* ***S2*** *= “cxzca”;*

***Output:*** *“bsxz”*

import java.util.HashSet; import java.util.Scanner; import java.util.Set;

public class UncommonCharacters { public static String concatenateUncommonCharacters(String s1, String s2) { Set<Character> uncommonChars = new HashSet<>();

for (char c : s1.toCharArray()) { if (!s2.contains(String.valueOf(c))) { uncommonChars.add(c);

}

}

for (char c : s2.toCharArray()) { if (!s1.contains(String.valueOf(c))) { uncommonChars.add(c);

}

}

StringBuilder result = new StringBuilder(); for (char c : uncommonChars) {

result.append(c);

}

return result.toString();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the first string: ");

String s1 = scanner.nextLine();

System.out.print("Enter the second string: ");

String s2 = scanner.nextLine();

String result = concatenateUncommonCharacters(s1, s2);

System.out.println("Result: " + result);

}

}

**B) Rearrange array such that even positioned are greater than odd**

Given an array A of n elements, sort the array according to the following relations :

* A[i]>=A[i-1], if i is even.
* A[i]<=A[i-1], if i is odd. Print the resultant array.

**Examples :**

Input : A[] = {1, 2, 2, 1}

Output : 1 2 1 2 Explanation :

For 1st element, 1 1, i = 2 is even.

3rd element, 1 1, i = 4 is even.

Input : A[] = {1, 3, 2}

Output : 1 3 2

Explanation :

Here, the array is also sorted as per the conditions.

1 1 and 2 < 3.

import java.util.Arrays; import java.util.Scanner; public class RearrangeArray { public static void rearrangeArray(int[] arr) {

int n = arr.length; Arrays.sort(arr); int left = 0; int right = n - 1; int[] result = new int[n]; for (int i = 0; i < n; i++) { if (i % 2 == 0) { result[i] = arr[right]; right--; } else { result[i] = arr[left]; left++;

} }

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

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

}

}

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[] arr = new int[n];

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

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

}

rearrangeArray(arr);

}

}

Q16)

**A) Maximum Consecutive Zeroes in Concatenated Binary String**

You are given a binary string **str** of length **n**. Suppose you create another string of size n \* k by concatenating **k** copies of str together. What is the maximum size of a substring of the concatenated string consisting only of 0’s? Given that k > 1. **Examples:**

***Input :*** *str = “110010”, k = 2*

***Output :*** *2*

*String becomes 110010110010 after two concatenations. This string has two zeroes.*

***Input :*** *str = “00100110”, k = 4*

***Output :*** *3*

**Complexity Analysis:**

***Time Complexity:* O(N)**, where N represents the length of the given string.

import java.util.Scanner; public class MaximumConsecutiveZeroes {

public static int getMaxConsecutiveZeroes(String str, int k) {

// Concatenate the string k times

StringBuilder sb = new StringBuilder();

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

sb.append(str);

}

int maxConsecutiveZeros = 0; int currentConsecutiveZeros = 0; for (int i = 0; i < sb.length(); i++) { if (sb.charAt(i) == '0') { currentConsecutiveZeros++;

maxConsecutiveZeros = Math.max(maxConsecutiveZeros, currentConsecutiveZeros);

} else {

currentConsecutiveZeros = 0;

}

}

return maxConsecutiveZeros;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the binary string: ");

String str = scanner.nextLine();

System.out.print("Enter the value of k: "); int k = scanner.nextInt();

int maxConsecutiveZeros = getMaxConsecutiveZeroes(str, k);

System.out.println("Maximum consecutive zeros: " + maxConsecutiveZeros);

scanner.close();

}

}

**B. Count smaller elements on Right side**

Given an unsorted array arr[] of distinct integers, construct another array countSmaller[] such that countSmaller[i] contains the count of smaller elements on the right side of each element arr[i] in the array.

**Examples:**

***Input:*** *arr[] = {12, 1, 2, 3, 0, 11, 4}*

***Output:*** *countSmaller[] = {6, 1, 1, 1, 0, 1, 0}*

***Input:*** *arr[] = {5, 4, 3, 2, 1}*

***Output:***  *countSmaller[] = {4, 3, 2, 1, 0}*

import java.util.Arrays; import java.util.Scanner; public class CountSmallerElements { public static void main(String[] args) { Scanner scanner = new Scanner(System.in); System.out.print("Enter the size of the array: "); int size = scanner.nextInt(); int[] arr = new int[size];

System.out.println("Enter the elements of the array:"); for (int i = 0; i < size; i++) { arr[i] = scanner.nextInt();

}

int[] countSmaller = new int[size]; for (int i = 0; i < size; i++) { int count = 0;

for (int j = i + 1; j < size; j++) {

if (arr[j] < arr[i]) { count++;

}

}

countSmaller[i] = count;

}

System.out.println("countSmaller[] = " + Arrays.toString(countSmaller));

}

}

17). A) Give a N\*N square matrix, return an array of its anti-diagonals. Look at the example for more details.

**Example:**

Input:

1 2 3

4 5 6

7 8 9

Return the following:

[

[1],

[2, 4],

[3, 5, 7],

[6, 8],

[9] ]

Input: 1 2

3 4

Return the following:

[

[1],

[2, 3],

[4]

]

import java.util.ArrayList; import java.util.Arrays; import java.util.List; import java.util.Scanner; public class AntiDiagonals { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); System.out.print("Enter the size of the square matrix: "); int size = scanner.nextInt(); int[][] matrix = new int[size][size];

System.out.println("Enter the elements of the matrix:"); for (int i = 0; i < size; i++) { for (int j = 0; j < size; j++) { matrix[i][j] = scanner.nextInt();

}

}

List<List<Integer>> antiDiagonals = getAntiDiagonals(matrix); System.out.println("Anti-Diagonals: "); for (List<Integer> diagonal : antiDiagonals) {

System.out.println(diagonal);

}

}

public static List<List<Integer>> getAntiDiagonals(int[][] matrix) { List<List<Integer>> antiDiagonals = new ArrayList<>(); int n = matrix.length; for (int i = 0; i < n; i++) {

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

int row = i; int col = 0; while (row >= 0 && col < n) { diagonal.add(matrix[row][col]);

row--; col++;

}

antiDiagonals.add(diagonal);

}

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

List<Integer> diagonal = new ArrayList<>(); int row = n - 1; int col = i; while (col < n) { diagonal.add(matrix[row][col]);

row--; col++;

}

antiDiagonals.add(diagonal);

}

return antiDiagonals;

}

}

**B) Expand the string according to the given conditions**

Given string **str** of the type **“3(ab)4(cd)”**, the task is to expand it to “abababcdcdcdcd” where integers are from the range **[1, 9]**.

**Examples:**

***Input:*** *str = “3(ab)4(cd)”*

***Output:*** *abababcdcdcdcd*

***Input:*** *str = “2(kl)3(ap)”*

***Output:*** *klklapapap*

import java.util.Stack; import java.util.Scanner; public class StringExpander { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the string: "); String str = scanner.nextLine();

String expandedStr = expandString(str);

System.out.println("Expanded String: " + expandedStr);

}

public static String expandString(String str) {

Stack<Integer> countStack = new Stack<>();

Stack<StringBuilder> stringStack = new Stack<>(); StringBuilder currentStr = new StringBuilder(); int currentCount = 0; for (char c : str.toCharArray()) { if (Character.isDigit(c)) { currentCount = currentCount \* 10 + (c - '0');

} else if (c == '(') { countStack.push(currentCount); stringStack.push(currentStr); currentCount = 0; currentStr = new StringBuilder();

} else if (c == ')') {

StringBuilder temp = currentStr; currentStr = stringStack.pop(); int count = countStack.pop(); for (int i = 0; i < count; i++) { currentStr.append(temp);

}

} else { currentStr.append(c);

}

}

return currentStr.toString();

}

}

18) A) Given an integer array **A** of size **N**. You need to check that whether there exist a element which is **strictly greater than all the elements on left** of it and **strictly smaller than all the elements** on right of it.If it exists return **1** else return **0**.

**NOTE:** Do not consider the corner elements i.e **A[0] and A[N-1]** as the answer.

**Problem Constraints:** 3 <= N <= 105

1 <= A[i] <= 109

**Input Format:** First and only argument is an integer array **A** containing **N** integers. **Output Format :** Return **1** if there exist a element that is **strictly greater than all the elements on left** of it and **strictly smaller than all the elements** on right of it else return 0. **Example Input**

Input 1: A = [5, 1, 4, 3, 6, 8, 10, 7, 9]

Input 2: A = [5, 1, 4, 4]

**Example Output:**

Output 1: 1

Output 2: 0

**Example Explanation**

Explanation 1: A[4] = 6 is the element we are looking for.

All elements on left of A[4] are smaller than it and all elements on right are greater.

Explanation 2: No such element exits.

import java.util.Scanner;

|  |
| --- |
| public class ElementCheck { 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[] A = new int[n];  System.out.println("Enter the elements:"); for (int i = 0; i < n; i++) { A[i] = scanner.nextInt();  } int result = checkElement(A);  System.out.println("Output: " + result);  } public static int checkElement(int[] A) { int n = A.length; int[] leftMax = new int[n]; int[] rightMin = new int[n]; leftMax[0] = A[0]; for (int i = 1; i < n; i++) { leftMax[i] = Math.max(leftMax[i - 1], A[i]);  } rightMin[n - 1] = A[n - 1]; for (int i = n - 2; i >= 0; i--) { rightMin[i] = Math.min(rightMin[i + 1], A[i]);  } for (int i = 1; i < n - 1; i++) { if (A[i] > leftMax[i - 1] && A[i] < rightMin[i + 1]) { |
| return 1;  } } return 0;  }  } |

B) GCD of more than two (or array) numbers

Given an array of numbers, find GCD of the array elements. In a previous post we [find GCD of two number.](https://www.geeksforgeeks.org/c-program-find-gcd-hcf-two-numbers/)

**Examples:**

Input : arr[] = {1, 2, 3}

Output : 1

Input : arr[] = {2, 4, 6, 8}

Output : 2

import java.util.Scanner; public class GCD {

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[] arr = new int[n];

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

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

}

int result = findGCD(arr);

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

}

public static int findGCD(int[] arr) { int gcd = arr[0]; for (int i = 1; i < arr.length; i++) { gcd = findGCD(gcd, arr[i]);

}

return gcd;

}

public static int findGCD(int a, int b) {

if (b == 0) { return a;

}

return findGCD(b, a % b);

}

}

19) A) Given a matrix of **M \* N** elements (M rows, N columns), return all elements of the matrix in spiral order.

**Problem Constraints**

1 <= **M, N** <= 1000

**Input Format**

The first argument is a matrix A.

**Output Format**

Return an array of integers representing all elements of the matrix in spiral order.

**Example Input**

A =

[

[ 1, 2, 3 ],

[ 4, 5, 6 ],

[ 7, 8, 9 ]

]

**Example Output**

[1, 2, 3, 6, 9, 8, 7, 4, 5]

import java.util.ArrayList; import java.util.List; import java.util.Scanner; public class SpiralMatrix { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); System.out.print("Enter the number of rows: "); int m = scanner.nextInt();

System.out.print("Enter the number of columns: "); int n = scanner.nextInt(); int[][] matrix = new int[m][n]; System.out.println("Enter the elements:"); for (int i = 0; i < m; i++) { for (int j = 0; j < n; j++) { matrix[i][j] = scanner.nextInt();

}

}

List<Integer> result = spiralOrder(matrix);

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

}

public static List<Integer> spiralOrder(int[][] matrix) { List<Integer> spiral = new ArrayList<>(); if (matrix == null || matrix.length == 0) {

return spiral;

}

int top = 0; int bottom = matrix.length - 1;

int left = 0;

int right = matrix[0].length - 1; while (top <= bottom && left <= right) {

for (int i = left; i <= right; i++) { spiral.add(matrix[top][i]);

}

top++; for (int i = top; i <= bottom; i++) { spiral.add(matrix[i][right]);

} right--;

if (top <= bottom) { // Traverse bottom row for (int i = right; i >= left; i--) { spiral.add(matrix[bottom][i]);

}

bottom--; }

if (left <= right) { for (int i = bottom; i >= top; i--) { spiral.add(matrix[i][left]);

} left++;

} }

return spiral;

}

}

B) Encrypt a string by repeating i-th character i times

Given string **str**, the task is to encrypt the string with the given encryption algorithm. The **1st** character of the string will be repeated **once** in the encrypted string, the **2nd** character will be repeated twice, …, **nth** character will be repeated **n** times.

**Examples:**

Input: str = "geeks"

Output: geeeeekkkksssss

Input: str = "abcd"

Output: abbcccdddd

import java.util.Scanner; public class StringEncryptor { public static String encrypt(String str) {

StringBuilder encryptedString = new StringBuilder();

for (int i = 0; i < str.length(); i++) { char c = str.charAt(i); for (int j = 0; j <= i; j++) { encryptedString.append(c);

}

}

return encryptedString.toString();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: ");

String str = scanner.nextLine();

String encryptedStr = encrypt(str);

System.out.println("Encrypted string: " + encryptedStr); scanner.close();

}

}

20)

**A) Find elements which are present in first array and not in second**

Given two arrays, the task is that we find numbers which are present in first array, but not present in the second array.

**Examples :**

Input : a[] = {1, 2, 3, 4, 5, 10}; b[] = {2, 3, 1, 0, 5};

Output : 4 10

4 and 10 are present in first array, but not in second array.

Input : a[] = {4, 3, 5, 9, 11}; b[] = {4, 9, 3, 11, 10};

Output : 5

import java.util.ArrayList; import java.util.List; import java.util.Scanner; public class ArrayDifference { public static List<Integer> findDifference(int[] arr1, int[] arr2) { List<Integer> difference = new ArrayList<>(); for (int num : arr1) { boolean found = false; for (int i : arr2) { if (num == i) { found = true; break;

} } if (!found) { difference.add(num);

}

} return difference;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); System.out.print("Enter the size of the first array: "); int size1 = scanner.nextInt(); int[] arr1 = new int[size1];

System.out.println("Enter the elements of the first array:"); for (int i = 0; i < size1; i++) { arr1[i] = scanner.nextInt();

}

System.out.print("Enter the size of the second array: "); int size2 = scanner.nextInt(); int[] arr2 = new int[size2];

System.out.println("Enter the elements of the second array:"); for (int i = 0; i < size2; i++) { arr2[i] = scanner.nextInt();

}

List<Integer> difference = findDifference(arr1, arr2);

System.out.println("Elements present in the first array but not in the second array: " + difference); scanner.close();

}

}

**B) Check if String formed by first and last X characters of a String is a Palindrome**

Given a [string](https://www.geeksforgeeks.org/category/data-structures/c-strings/) **str** and an integer **X**. The task is to find whether the first **X** characters of both [string](https://www.geeksforgeeks.org/category/data-structures/c-strings/) **str** and [reversed](https://www.geeksforgeeks.org/reverse-a-string-in-java/) [string](https://www.geeksforgeeks.org/category/data-structures/c-strings/) **str** are same or not. If it is equal then print **true**, otherwise print **false**.

**Examples:**

***Input****: str = abcdefba, X = 2* ***Output****: true*

***Explanation****:*

*First 2 characters of both string* ***str*** *and reversed string* ***str*** *are same.*

***Input****: str = GeeksforGeeks, X = 3*

***Output****: false*

import java.util.Scanner; public class PalindromeChecker { public static boolean isPalindrome(String str, int X) { String firstXChars = str.substring(0, X);

String lastXChars = str.substring(str.length() - X);

String reversedLastXChars = new StringBuilder(lastXChars).reverse().toString(); return firstXChars.equals(reversedLastXChars);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: ");

String str = scanner.nextLine();

System.out.print("Enter the value of X: "); int X = scanner.nextInt(); boolean result = isPalindrome(str, X); System.out.println(result); scanner.close();

}

}

**21) A) Check if array elements are consecutive. 30M**

Given an unsorted array of numbers, write a function that returns true if the array consists of consecutive numbers.

Test Cases:

1. If the array is {5, 2, 3, 1, 4}, then the function should return true because the array has consecutive numbers from 1 to 5.

1. If the array is {83, 78, 80, 81, 79, 82}, then the function should return true because the array has consecutive numbers from 78 to 83.

1. If the array is {34, 23, 52, 12, 3}, then the function should return false because the elements are not consecutive.

1. If the array is {7, 6, 5, 5, 3, 4}, then the function should return false because 5 and 5 are not consecutive.

Expected **Time Complexity:** O(n)

**Auxiliary Space:** O(n)

For Time Complexity O(nlogn): 15 Marks

For Time Complexity O(n): 30 Marks

import java.util.Arrays; import java.util.Scanner; public class ConsecutiveArrayChecker { public static boolean areElementsConsecutive(int[] array) { int min = Arrays.stream(array).min().getAsInt(); int max = Arrays.stream(array).max().getAsInt(); int n = array.length; if (max - min + 1 != n) { return false;

}

boolean[] visited = new boolean[n]; for (int i = 0; i < n; i++) { int num = array[i] - min; if (visited[num]) { return false;

}

visited[num] = true;

} return true;

}

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[] array = new int[n];

System.out.println("Enter the elements of the array:"); for (int i = 0; i < n; i++) { array[i] = scanner.nextInt();

}

System.out.println("Result: " + areElementsConsecutive(array)); scanner.close();

}

}

B) Check if a given string is a rotation of a palindrome 30 M

Given a string, check if it is a rotation of a palindrome. For example your function should return true for “aab” as it is a rotation of “aba”.

**Examples:**

Input: str = "aaaad"

Output: 1

// "aaaad" is a rotation of a palindrome "aadaa"

Input: str = "abcd"

Output: 0

// "abcd" is not a rotation of any palindrome. import java.util.Scanner; public class RotationPalindromeChecker { public static boolean isRotationOfPalindrome(String str) {

if (isPalindrome(str)) {

return true;

}

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

String rotatedStr = str.substring(i) + str.substring(0, i); if (isPalindrome(rotatedStr)) {

return true;

} }

return false;

}

public static boolean isPalindrome(String str) {

int n = str.length(); for (int i = 0, j = n - 1; i < j; i++, j--) { if (str.charAt(i) != str.charAt(j)) { return false;

} }

return true;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the string: ");

String str = scanner.nextLine();

System.out.println("Result: " + isRotationOfPalindrome(str));

scanner.close();

}

}

**22) A) Count Distinct Strings present in an array**

Given an [array of strings](https://www.geeksforgeeks.org/array-strings-c-3-different-ways-create/) **arr[]**, the task is to find the count of distinct strings present in the array.

**Examples:**

***Input:*** *arr[] = { “abcde”, “abcce”, “abcdf”, “abcde”, “abcdf” }* ***Output:*** *3*

***Explanation:***

*Distinct strings in the array are { “abcde”, “abcce”, “abcdf” }. Therefore, the required output is 3.*

***Input:*** *arr[] = { “ab”, “abc”, “abcd”, “abcde”, “a” }* ***Output:*** *5*

***Explanation:***

*Distinct strings in the array are { “abcde”, “abcd”, “abc”, “ab”, “a” }.*

*Therefore, the required output is 5.*

import java.util.HashSet; import java.util.Scanner; public class DistinctStringCounter { public static int countDistinctStrings(String[] arr) { HashSet<String> distinctStrings = new HashSet<>();

for (String str : arr) { distinctStrings.add(str);

}

return distinctStrings.size();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); System.out.print("Enter the number of strings: "); int n = scanner.nextInt(); scanner.nextLine(); // Consume the newline character

String[] arr = new String[n];

System.out.println("Enter the strings:"); for (int i = 0; i < n; i++) { arr[i] = scanner.nextLine();

}

System.out.println("Distinct strings count: " + countDistinctStrings(arr)); scanner.close();

}

}

**B) Count of elements in Array which are present K times & their double isn’t present**

Given an [array](https://www.geeksforgeeks.org/array-data-structure/) **arr[]** of N integers, the task is to find the count of elements in the array that are present **K times** and their double are not present in the array.

**Examples:**

***Input:*** *arr[] = {10, 6, 12, 8, 10, 8}, K = 2*

***Output:*** *2*

***Explanation:*** *10 is a valid number since it appears exactly two times and 20 does not appear in array.*

*8 is a valid number since it appears two times and 16 does not appear in array.* ***Input:*** *arr[] = {1, 3, 5, 3}, K = 3*

***Output:*** *0*

***Explanation:*** *No element in the given array satisfy the condition.*

*Input: arr[]={1,3,5,3,4,3} K=3*

***Output****:0*

|  |  |  |
| --- | --- | --- |
| ***Input:*** *arr[] = {1, 2, 2, 3, 3, 4}, K = 2* | |  |
| ***Output:*** *1* |  |
| ***Explanation:*** *Only 3 is valid element.* | |
| *Though 2 is present twice but its double is also present.* | | |

import java.util.HashMap; import java.util.Map; import java.util.Scanner; public class CountElements { public static int countElements(int[] arr, int K) { int count = 0;

Map<Integer, Integer> elementCount = new HashMap<>();

for (int i = 0; i < arr.length; i++) { elementCount.put(arr[i], elementCount.getOrDefault(arr[i], 0) + 1);

} for (int i = 0; i < arr.length; i++) { int element = arr[i]; int doubleElement = 2 \* element;

if (elementCount.getOrDefault(element, 0) == K &&

!elementCount.containsKey(doubleElement)) {

count++;

}

} return count;

}

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[] arr = new int[n];

System.out.println("Enter the elements of the array:"); for (int i = 0; i < n; i++) { arr[i] = scanner.nextInt();

}

System.out.print("Enter the value of K: "); int K = scanner.nextInt(); int result = countElements(arr, K);

System.out.println("Count of elements: " + result); scanner.close();

}

}

***23) A)* Count of strings with frequency of each character at most K**

Given an [array](https://www.geeksforgeeks.org/array-data-structure/) **arr[]** containing **N** [strings](http://www.geeksforgeeks.org/frequent-word-array-strings/) and an integer **K**, the task is to find the count of [strings](https://www.geeksforgeeks.org/category/data-structures/c-strings/) with the frequency of each character at most **K** **Examples:**

***Input:*** *arr[] = { “abab”, “derdee”, “erre” }, K = 2* ***Output:*** *2*

***Explanation:*** *Strings with character frequency at most 2 are “abab”, “erre”. Hence count is 2*

***Input:*** *arr[] = {“ag”, “ka”, “nanana”}, K = 3* ***Output:*** *1* import java.util.HashMap; import java.util.Map; import java.util.Scanner; public class StringCount { public static int countStrings(String[] arr, int K) { int count = 0; for (String s : arr) {

Map<Character, Integer> frequencyMap = new HashMap<>();

for (char c : s.toCharArray()) { frequencyMap.put(c, frequencyMap.getOrDefault(c, 0) + 1);

}

boolean isBelowK = true; for (int freq : frequencyMap.values()) {

if (freq > K) { isBelowK = false;

break;

}

}

if (isBelowK) { count++;

}

}

return count;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); System.out.print("Enter the number of strings: "); int numStrings = scanner.nextInt();

String[] arr = new String[numStrings]; System.out.println("Enter the strings:"); for (int i = 0; i < numStrings; i++) { arr[i] = scanner.next();

}

System.out.print("Enter the value for K: "); int K = scanner.nextInt(); scanner.close();

int result = countStrings(arr, K);

System.out.println("Count of strings: " + result);

}

}

***B)* Count all elements in the array which appears at least K times after their first occurrence**

Given an array **arr[]** of **N** integer elements and an integer **K**. The task is to count all distinct **arr[i]** such that **arr[i]** appears at least **K** times in the index range **i + 1** to **n – 1**.

**Examples:**

***Input:*** *arr[] = {1, 2, 1, 3}, K = 1*

***Output:*** *1*

*arr[0] = 1 is the only element that appears at least once in the index range [1, 3] i.e. arr[2]*

***Input:*** *arr[] = {1, 2, 3, 2, 1, 3, 1, 2, 1}, K = 2* ***Output:*** *2*

import java.util.HashMap; import java.util.Map; import java.util.Scanner; public class CountElements {

public static int countElements(int[] arr, int K) { int count = 0; for (int i = 0; i < arr.length; i++) { int element = arr[i]; int frequency = 0;

Map<Integer, Integer> frequencyMap = new HashMap<>();

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

frequencyMap.put(arr[j], frequencyMap.getOrDefault(arr[j], 0) + 1);

}

if (frequencyMap.containsKey(element) && frequencyMap.get(element) >= K) {

count++;

}

}

return count;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of elements in the array: "); int numElements = scanner.nextInt(); int[] arr = new int[numElements]; System.out.println("Enter the elements:"); for (int i = 0; i < numElements; i++) { arr[i] = scanner.nextInt();

}

System.out.print("Enter the value for K: "); int K = scanner.nextInt(); scanner.close(); int result = countElements(arr, K);

System.out.println("Count of elements: " + result);

}

}

***24) A)* Print all characters of string whose frequency is a power of K**

Given [string](https://www.geeksforgeeks.org/string-class-in-java/) **str** of size **N**, the task is to print the characters of string whose frequency is a power of **K** in a lexicographically sorted order. **Examples:**

***Input:*** *str = “aaacbb” K = 2*

***Output:*** *bbc*

***Explanation:*** *Frequency of a is 3 which is not the power of 2. Frequency of c is 1 and frequency of b is 2 which are the power of 2.*

***Input:*** *str = “geeksgeekgeeks” K = 3*

***Output:*** *eeeeeegggkkk*

import java.util.\*; public class FrequencyPowerK { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: ");

String str = scanner.nextLine();

System.out.print("Enter the value of K: "); int K = scanner.nextInt(); printCharactersWithFrequencyPowerK(str, K); scanner.close();

}

private static void printCharactersWithFrequencyPowerK(String str, int K) {

Map<Character, Integer> frequencyMap = new HashMap<>(); List<Character> result = new ArrayList<>(); for (char ch : str.toCharArray()) { frequencyMap.put(ch, frequencyMap.getOrDefault(ch, 0) + 1);

}

for (char ch : frequencyMap.keySet()) { int frequency = frequencyMap.get(ch); if (isPowerOfK(frequency, K)) { result.add(ch);

}

}

Collections.sort(result); for (char ch : result) {

System.out.print(ch);

}

}

private static boolean isPowerOfK(int num, int K) { while (num > 1 && num % K == 0) {

num /= K;

}

return num == 1;

}

}

***B)* Find all matrix elements which are minimum in their row and maximum in their column**

Given a matrix **mat[][]** of size **M \* N**, the task is to find all matrix elements which are minimum in their respective row and maximum in their respective column. If no such element is present, print **-1**.

**Examples:**

***Input:*** *mat[][] = {{1, 10, 4}, {9, 3, 8}, {15, 16, 17}}*

***Output:*** *15* ***Explanation:***

*15 is the only element which is maximum in its column {1, 9,* ***15****} and minimum in its row {****15****, 16, 17}.*

***Input:*** *m[][] = {{10, 41}, {3, 5}, {16, 2}}*

***Output:*** *-1* import java.util.\*; public class MatrixMinMax { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); System.out.print("Enter the number of rows: "); int rows = scanner.nextInt();

System.out.print("Enter the number of columns: "); int columns = scanner.nextInt(); int[][] mat = new int[rows][columns];

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

for (int i = 0; i < rows; i++) { for (int j = 0; j < columns; j++) { mat[i][j] = scanner.nextInt();

}

}

findMatrixElements(mat); scanner.close();

}

private static void findMatrixElements(int[][] mat) { int rows = mat.length; int columns = mat[0].length; List<Integer> result = new ArrayList<>(); for (int i = 0; i < rows; i++) { int min = Integer.MAX\_VALUE; int minColumnIndex = -1; for (int j = 0; j < columns; j++) { if (mat[i][j] < min) { min = mat[i][j]; minColumnIndex = j;

}

}

boolean isMaximumInColumn = true; for (int k = 0; k < rows; k++) { if (mat[k][minColumnIndex] > min) { isMaximumInColumn = false; break;

}

}

if (isMaximumInColumn) {

result.add(min);

}

}

if (result.isEmpty()) {

System.out.println("-1");

} else {

for (int num : result) {

System.out.print(num + " ");

}

}

}

}

***25) A)* Replace the odd positioned elements with their cubes and even positioned elements with their squares**

Given an array **arr[]** of **n** elements, the task is to replace all the odd positioned elements with their cubes and even positioned elements with their squares i.e. the resultant array must be **{arr[0]3, arr[1]2, arr[2]3, arr[3]2, …}**.

**Examples:**

***Input:*** *arr[]= {2, 3, 4, 5}*

***Output:*** *8 9 64 25*

*Updated array will be {23, 32, 43, 52} -> {8, 9, 64, 25}*

***Input:*** *arr[] = {3, 4, 5, 2}*

***Output:*** *27 16 125 4*

import java.util.Scanner; public class ArrayManipulation { public static void main(String[] args) { Scanner scanner = new Scanner(System.in); System.out.print("Enter the size of the array: "); int size = scanner.nextInt(); int[] arr = new int[size];

System.out.println("Enter the elements of the array:"); for (int i = 0; i < size; i++) { arr[i] = scanner.nextInt();

}

int[] updatedArr = manipulateArray(arr); System.out.println("Updated array:"); for (int num : updatedArr) {

System.out.print(num + " ");

} scanner.close();

}

public static int[] manipulateArray(int[] arr) { for (int i = 0; i < arr.length; i++) { if (i % 2 == 0) { arr[i] = arr[i] \* arr[i];

} else {

arr[i] = arr[i] \* arr[i] \* arr[i];

} } return arr;

}

}

*B)* Given a string s, reverse only all the vowels in the string and return it.

The vowels are 'a', 'e', 'i', 'o', and 'u', and they can appear in both lower and upper cases, more than once.

Example 1:

Input: s= "hello" Output: "holle" Example 2:

Input: s= "AEIOU" Output: "UOIEA" Example 3:

Input: s= "DesignGUrus"

Output: "DusUgnGires"

public class ReverseVowels { public static String reverseVowels(String s) {

char[] vowels = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'}; char[] charArray = s.toCharArray();

int i = 0; int j = s.length() - 1; while (i < j) { if (isVowel(charArray[i], vowels) && isVowel(charArray[j], vowels)) { char temp = charArray[i]; charArray[i] = charArray[j]; charArray[j] = temp; i++;

j--;

} else if (!isVowel(charArray[i], vowels)) {

i++; } else { j--;

}

}

return new String(charArray);

}

public static boolean isVowel(char c, char[] vowels) { for (char vowel : vowels) { if (c == vowel) { return true;

} } return false;

}

public static void main(String[] args) {

String s = "hello";

System.out.println(reverseVowels(s)); s = "AEIOU";

System.out.println(reverseVowels(s)); s = "DesignGUrus";

System.out.println(reverseVowels(s));

}

*26) A)* Given an array of strings words and two different strings that already exist in the array word1 and word2, return the shortest distance between these two words in the list.

Example 1:

Input: words = ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"], word1 = "fox", word2 = "dog"

Output: 5

Explanation: The distance between "fox" and "dog" is 5 words.

Example 2:

Input: words = ["a", "c", "d", "b", "a"], word1 = "a", word2 = "b"

Output: 1

Explanation: The shortest distance between "a" and "b" is 1 word

import java.util.Scanner; public class ShortestDistance { public static int shortestDistance(String[] words, String word1, String word2) { int minDistance = Integer.MAX\_VALUE;

int index1 = -1; int index2 = -1; for (int i = 0; i < words.length; i++) { if (words[i].equals(word1)) { index1 = i;

} else if (words[i].equals(word2)) { index2 = i;

}

if (index1 != -1 && index2 != -1) { minDistance = Math.min(minDistance, Math.abs(index1 - index2));

}

}

return minDistance;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); System.out.print("Enter the number of words: "); int numWords = scanner.nextInt(); scanner.nextLine();

String[] words = new String[numWords]; for (int i = 0; i < numWords; i++) {

System.out.print("Enter word " + (i+1) + ": "); words[i] = scanner.nextLine();

}

System.out.print("Enter word 1: ");

String word1 = scanner.nextLine(); System.out.print("Enter word 2: "); String word2 = scanner.nextLine(); scanner.close(); int shortestDist = shortestDistance(words, word1, word2);

System.out.println("Shortest distance: " + shortestDist);

}

}

*B)* Given an array of integers nums, return the number of good pairs.

A pair (i, j) is called good if nums[i] == nums[j] and i < j.

Example 1:

Input: nums = [1,2,3,1,1,3]

Output: 4

Explanation: There are 4 good pairs, here are the indices: (0,3), (0,4), (3,4), (2,5).

Example 2:

Input: nums = [1,1,1,1]

Output: 6

Explanation: Each pair in the array is a 'good pair'.

Example 3:

Input: words = nums = [1,2,3]

Output: 0

Explanation: No number is repeating.

import java.util.Scanner; public class GoodPairs {

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

System.out.print("Enter element " + (i + 1) + ": "); nums[i] = scanner.nextInt();

}

scanner.close();

int result = numIdenticalPairs(nums);

System.out.println("Number of good pairs: " + result);

}

public static int numIdenticalPairs(int[] nums) { int count = 0;

for (int i = 0; i < nums.length - 1; i++) { for (int j = i + 1; j < nums.length; j++) { if (nums[i] == nums[j] && i < j) {

count++;

}

}

}

return count;

}

}

***27) A)* Check if all given strings are isograms or not**

Given an array **arr** containing **N** strings, the task is to check if all strings are [isogram](https://www.geeksforgeeks.org/check-string-isogram-not/) or not. If they are, print **Yes**, otherwise **No**.

*An* ***Isogram*** *is a word in which no letter occurs more than once.* **Examples:**

***Input:*** *arr[] = {“abcd”, “derg”, “erty”}*

|  |  |
| --- | --- |
| ***Input:*** *arr[] = {“agka”, “lkmn”}* | |
| ***Output:*** *No* |  |

***Output:*** *Yes*

import java.util.HashSet; import java.util.Scanner; public class IsogramCheck { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); System.out.print("Enter the number of strings: "); int n = scanner.nextInt(); scanner.nextLine(); String[] arr = new String[n]; for (int i = 0; i < n; i++) {

System.out.print("Enter string " + (i + 1) + ": "); arr[i] = scanner.nextLine();

} scanner.close(); boolean result = checkIsograms(arr);

System.out.println(result ? "Yes" : "No");

}

public static boolean checkIsograms(String[] arr) {

for (String str : arr) {

HashSet<Character> set = new HashSet<>(); for (char ch : str.toCharArray()) { if (set.contains(ch)) { return false;

} set.add(ch);

} } return true;

}

}

***B)* Sum of all odd frequency elements in a Matrix**

Given a NxM matrix of integers containing duplicate elements. The task is to find the sum of all odd occurring elements in the given matrix. That is the sum of all such elements whose frequency is odd in the matrix.

**Examples**:

**Input** : mat[] = {{1, 1, 2},

{2, 3, 3},

{4, 5, 3}}

## Output : 18

The odd occurring elements are 3, 4, 5 and their number of occurrences are 3, 1, 1 respectively. Therefore, sum = 3+3+3+4+5 = 18.

**Input** : mat[] = {{10, 20},

{40, 40}}

## Output : 30

import java.util.HashMap; import java.util.Map; import java.util.Scanner; public class MatrixSum {

public static int calculateSum(int[][] matrix) { Map<Integer, Integer> frequencyMap = new HashMap<>(); int sum = 0; for (int[] row : matrix) { for (int num : row) { frequencyMap.put(num, frequencyMap.getOrDefault(num, 0) + 1);

} } for (Map.Entry<Integer, Integer> entry : frequencyMap.entrySet()) {

int element = entry.getKey(); int frequency = entry.getValue(); if (frequency % 2 != 0) { sum += element;

} } return sum;

} public static void main(String[] args) { Scanner scanner = new Scanner(System.in); System.out.print("Enter the number of rows: "); int rows = scanner.nextInt();

System.out.print("Enter the number of columns: "); int columns = scanner.nextInt(); int[][] matrix = new int[rows][columns]; System.out.println("Enter the matrix elements:"); for (int i = 0; i < rows; i++) { for (int j = 0; j < columns; j++) { matrix[i][j] = scanner.nextInt();

} }

int sum = calculateSum(matrix);

System.out.println("Sum of all odd occurring elements: " + sum);

scanner.close();

}

}

|  |  |
| --- | --- |
| ***28) A)*** | **Modify a string by circularly shifting each character to the right by respective** |
| **frequencies** | |

Given a [string](https://www.geeksforgeeks.org/string-data-structure/) **S** consisting of lowercase English alphabets, the task is to right shift each character of the given string **S** circularly by its frequency.

*Circular shifting of characters refers to shifting character ‘z’ to ‘a’, as its next character.*

**Examples:**

***Input:*** *S = “geeksforgeeks”* ***Output:*** *iiimugpsiiimu* ***Explanation:***

*Following changes are made on the string S:*

1. *Frequency of ‘g’ is 2. Therefore, shifting the character ‘g’ by 2 becomes ‘i’.*
2. *Frequency of ‘e’ is 4. Therefore, shifting the character ‘e’ by 4 becomes ‘i’.*
3. *Frequency of ‘k’ is 2. Therefore, shifting the character ‘k’ by 2 becomes ‘m’.*
4. *Frequency of ‘s’ is 2. Therefore, shifting the character ‘s’ by 2 becomes ‘u’.*
5. *Frequency of ‘f’ is 1. Therefore, shifting the character ‘f’ by 1 becomes ‘g’.*
6. *Frequency of ‘o’ is 1. Therefore, shifting the character ‘o’ by 1 becomes ‘p’.*
7. *Frequency of ‘r’ is 1. Therefore, shifting the character ‘r’ by 1 becomes ‘s’. After the above shifting of characters, the string modifies to “iiimugpsiiimu”.*

***Input:*** *S = “aabcadb”*

***Output:*** *ddddded*

import java.util.Scanner; public class CircularShift { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: "); String S = scanner.nextLine(); int[] freq = new int[26]; for (char c : S.toCharArray()) { freq[c - 'a']++;

}

StringBuilder sb = new StringBuilder(); for (char c : S.toCharArray()) { int shift = freq[c - 'a']; char shiftedChar = (char) (((c - 'a' + shift) % 26) + 'a'); sb.append(shiftedChar);

}

String modifiedString = sb.toString();

System.out.println("Modified string: " + modifiedString);

}

}

***B)* Print matrix in snake pattern**

Given an n x n matrix. In the given matrix, you have to print the elements of the matrix in the snake pattern.

**Examples :**

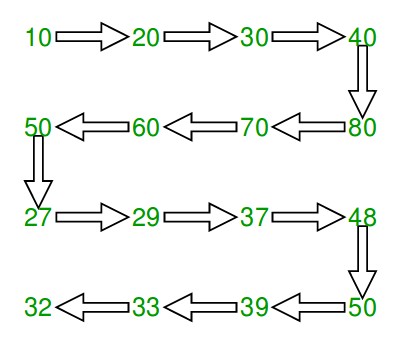
***Input:*** *mat[][] = { {10, 20, 30, 40},*

*{15, 25, 35, 45},*

*{27, 29, 37, 48},*

*{32, 33, 39, 50}};*

***Output:*** *10 20 30 40 45 35 25 15 27 29 37 48 50 39 33 32*



import java.util.Scanner; public class SnakePattern { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of rows: "); int rows = scanner.nextInt();

System.out.print("Enter the number of columns: "); int cols = scanner.nextInt(); int[][] matrix = new int[rows][cols];

System.out.println("Enter the matrix elements:"); for (int i = 0; i < rows; i++) { for (int j = 0; j < cols; j++) { matrix[i][j] = scanner.nextInt();

}

} for (int i = 0; i < rows; i++) { if (i % 2 == 0) { for (int j = 0; j < cols; j++) {

System.out.print(matrix[i][j] + " ");

} } else { for (int j = cols - 1; j >= 0; j--) {

System.out.print(matrix[i][j] + " ");

}

}

}

}

}

***29) A)* Program to check if all characters have even frequency**

Given a string S consisting only of lowercase letters check if the string has all characters appearing even times.

**Examples:**

***Input :*** *abaccaba*

***Output :*** *Yes*

***Explanation:*** *‘a’ occurs four times, ‘b’ occurs twice, ‘c’ occurs twice and the other letters occur zero times.*

***Input:***  *hthth* ***Output :*** *No*

import java.util.HashMap; import java.util.Map; import java.util.Scanner; public class EvenFrequencyChecker { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: ");

String S = scanner.nextLine();

Map<Character, Integer> frequencyMap = new HashMap<>(); for (char c : S.toCharArray()) { frequencyMap.put(c, frequencyMap.getOrDefault(c, 0) + 1);

}

boolean allEvenFrequency = true; for (int freq : frequencyMap.values()) { if (freq % 2 != 0) { allEvenFrequency = false; break;

}

}

if (allEvenFrequency) {

System.out.println("yes");

} else {

System.out.println("No");

}

}

}

***B)* Squares of Matrix Diagonal Elements**

You have given an integer matrix with odd dimensions. Find the square of the diagonal elements on both sides.

**Examples:**

Input : 1 2 3

1. 5 6

7 8 9

Output : Diagonal one: 1 25 81

Diagonal two: 9 25 49

Input : 2 5 7

3 7 2

1. 6 9

Output : Diagonal one : 4 49 81 Diagonal two : 49 49 25 import java.util.Scanner; public class MatrixDiagonalSquares { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of rows in the matrix:

");

int rows = scanner.nextInt();

System.out.print("Enter the number of columns in the matrix: ");

int columns = scanner.nextInt(); int[][] matrix = new int[rows][columns];

System.out.println("Enter the elements of the matrix:"); for (int i = 0; i < rows; i++) { for (int j = 0; j < columns; j++) { matrix[i][j] = scanner.nextInt();

} }

int size = rows;

int[] diagonalSquares = new int[size]; for (int i = 0; i < size; i++) {

diagonalSquares[i] = matrix[i][i] \* matrix[i][i]; diagonalSquares[i] += matrix[i][size - i - 1] \* matrix[i][size - i - 1];

}

System.out.println("Squares of diagonal elements on both sides:");

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

System.out.println(diagonalSquares[i]);

}

}

}

***30) A)* Count of strings that does not contain any character of a given string**

Given an array **arr** containing **N** strings and a string **str**, the task is to find the number of strings that do not contain any character of string **str**. **Examples:**

***Input:*** *arr[] = {“abcd”, “hijk”, “xyz”, “ayt”}, str=”apple”*

***Output:*** *2*

***Explanation:*** *“hijk” and “xyz” are the strings that do not contain any character of str*

***Input:*** *arr[] = {“apple”, “banana”, “pear”}, str=”nil”*

***Output:*** *1*

import java.util.Scanner; public class StringCount {

public static int countStrings(String[] arr, String str) { int count = 0; for (String s : arr) {

if (!containsAnyCharacter(s, str)) { count++;

}

}

return count;

}

public static boolean containsAnyCharacter(String s, String str) { for (char c : str.toCharArray()) { if (s.contains(String.valueOf(c))) { return true;

}

} return false;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int n = scanner.nextInt(); scanner.nextLine(); String[] arr = new String[n]; for (int i = 0; i < n; i++) {

System.out.print("Enter string " + (i + 1) + ": "); arr[i] = scanner.nextLine();

}

System.out.print("Enter the string to check: "); String str = scanner.nextLine(); int result = countStrings(arr, str);

System.out.println("Number of strings that do not contain any character of " + str + ": " + result); scanner.close();

}

}

*B)* Given an m x n matrix, find all common elements present in all rows in O(mn) time and one traversal of matrix.

**Example:**

**Input:**

mat[4][5] = {{1, 2, 1, 4, 8}, {3, 7, 8, 5, 1},

{8, 7, 7, 3, 1},

{8, 1, 2, 7, 9},

};

**Output:**

1 8 or 8 1

8 and 1 are present in all rows.

import java.util.ArrayList; import java.util.HashMap; import java.util.List; import java.util.Map; import java.util.Scanner; public class CommonElementsInMatrix { public static List<Integer> findCommonElements(int[][] matrix) { List<Integer> commonElements = new ArrayList<>(); if (matrix.length == 0 || matrix[0].length == 0) { return commonElements;

}

Map<Integer, Integer> countMap = new HashMap<>();

for (int j = 0; j < matrix[0].length; j++) { countMap.put(matrix[0][j], 1);

}

for (int i = 1; i < matrix.length; i++) { for (int j = 0; j < matrix[i].length; j++) { int element = matrix[i][j]; if (countMap.containsKey(element) && countMap.get(element) == i) { countMap.put(element, i + 1);

}

}

}

for (Map.Entry<Integer, Integer> entry : countMap.entrySet()) { if (entry.getValue() == matrix.length) { commonElements.add(entry.getKey());

}

}

return commonElements;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); System.out.print("Enter the number of rows: "); int rows = scanner.nextInt();

System.out.print("Enter the number of columns: "); int columns = scanner.nextInt(); int[][] matrix = new int[rows][columns];

System.out.println("Enter the elements of the matrix:"); for (int i = 0; i < rows; i++) { for (int j = 0; j < columns; j++) { matrix[i][j] = scanner.nextInt();

}

}

List<Integer> commonElements = findCommonElements(matrix); System.out.println("Common Elements: " + commonElements);

scanner.close();

}

}

***31) A)* Count of sticks required to represent the given string**

Given a string **str** of uppercase alphabets and numbers, the task is to find the number of matchsticks required to represent it.

***Input:***

*str = “ABC2”*

***Output:***

*22*

***Explanatio***

***n:***

*quired to represent A,*

*re*

*6*

*sticks are*

*quired to represent B,*

*sticks are*

*7*

*re*

*sticks are*

*4*

*re*

*quired to represent C,*

*quired to represent*

*2.*

*5*

*sticks are*

*re*

*Therefore t*

*he*

*total number of match*

*sticks required is 6 + 7 + 4 + 5 = 22.*

***Input:***

*“*

*GEEKSFORGEEKS”*

*str*

*=*

***Output:***

*66*

***Explanation:***

*6*

*sticks are required to represent G,*

*5*

*sticks are required to represent E,*

*4*

*sticks are required to represent K,*

*5*

*sticks are required to represent S,*

*4*

*sticks are required to represent F,*

*6*

*sticks are required to represent O,*

*6*

*sticks are required to represent R.*

*Therefore the total number of matchsticks re*

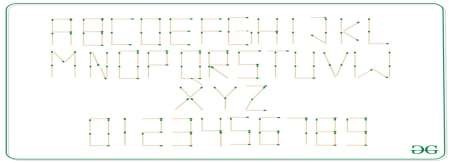
*quired is 6 + 5 + 5 + 4 + 5 + 4 + 6 + 6 + 6 +*

*5*

*+ 5 + 4 + 5 =*

*17*

*.*



import java.util.Scanner; public class MatchstickCounter { public static int countMatchsticks(String str) { int count = 0; for (int i = 0; i < str.length(); i++) { char ch = str.charAt(i); if (Character.isAlphabetic(ch)) { count += getMatchstickCount(ch); } else if (Character.isDigit(ch)) {

int num = Character.getNumericValue(ch); count += getMatchstickCount(num);

}

}

return count;

}

private static int getMatchstickCount(char ch) { switch (ch) { case 'A': return 2; case 'B': return 3; default: return 0;

}

}

private static int getMatchstickCount(int num) { return num \* 6;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the string: "); String str = scanner.nextLine(); scanner.close();

int matchstickCount = countMatchsticks(str);

|  |  |
| --- | --- |
| System.out.println("Number of matchsticks required: " + | |
| matchstickCount); |  |

}

}

|  |  |
| --- | --- |
| ***B)*** | **Maximum difference between first and last indexes of an element in array** |
| Given an array of n integers. The task is to find the difference of first and last index of each | |

distinct element so as to maximize the difference.

**Examples:**

Input : {2, 1, 3, 4, 2, 1, 5, 1, 7}

Output : 6

Element **1** has its **first index** = 1 and **last index** = 7 **Difference** = 7 - 1 = 6

Other elements have a smaller first and last index difference

Input : {2, 2, 1, 1, 8, 8, 3, 5, 3}

Output : 2

Maximum difference is for indexes of element 3.

import java.util.HashMap; import java.util.Scanner; public class Main { public static void main(String[] args) { Scanner scanner = new Scanner(System.in); System.out.print("Enter the size of the array: "); int n = scanner.nextInt(); int[] arr = new int[n];

System.out.println("Enter the elements of the array:"); for (int i = 0; i < n; i++) { arr[i] = scanner.nextInt();

}

HashMap<Integer, Integer> firstIndexMap = new HashMap<>(); HashMap<Integer, Integer> lastIndexMap = new HashMap<>(); int maxDifference = 0; for (int i = 0; i < arr.length; i++) { int num = arr[i];

if (!firstIndexMap.containsKey(num)) {

firstIndexMap.put(num, i);

}

lastIndexMap.put(num, i);

maxDifference = Math.max(maxDifference, lastIndexMap.get(num) - firstIndexMap.get(num));

}

System.out.println("Max difference of first and last index: " + maxDifference); }

}

***32) A)* Most similar string**

Given a string **str** and an [array](https://www.geeksforgeeks.org/introduction-to-arrays/) of strings **arr[]** of size **N**, the task is to print a string from arr[], which has maximum count of matching characters with str. **Examples:**

***Input:*** *str = “vikas”, N = 3, arr[] = [“preeti”, “khusbu”, “katherina”]* ***Output:*** *“katherina”* ***Explanation:***

*Number of similar characters between Str and each string in D[ ] are,*

*“preeti” = 1*

*“khusbu” = 2*

*“katherina” = 3*

*Hence, “katherina” has maximum matching characters.* ***Input:*** *str = “gfg”, N = 3, arr[] = [“goal”, “fog”, “abc”]* ***Output:*** *“fog”* ***Explanation:***

*Number of similar characters between Str and each string in D[ ] are,*

*“goal” = 1*

*“fog” = 2*

*“abc” = 0*

*Hence, “fog” has maximum matching characters.*

*import java.util.HashMap; import java.util.Scanner;* *public class Main {*  *public static void main(String[] args) {*  *Scanner scanner = new Scanner(System.in);*

*System.out.print("Enter the string: ");*

*String str = scanner.nextLine();*

*System.out.print("Enter the number of strings: ");*  *int n = scanner.nextInt();*  *scanner.nextLine();*

*String[] arr = new String[n];*

*System.out.println("Enter the strings:");*

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

*arr[i] = scanner.nextLine();*

*}*

*String result = "";*  *int maxCount = 0;*  *for (String s : arr) {*

*int count = countMatchingCharacters(str, s);*  *if (count > maxCount) {*  *maxCount = count;*

*result = s;*

*}*

*}*

*System.out.println("String with maximum matching characters: " + result);*

*}*

*private static int countMatchingCharacters(String str1, String str2) {*

*HashMap<Character, Integer> charCountMap = new HashMap<>();*  *for (char c : str1.toCharArray()) {*

*charCountMap.put(c, charCountMap.getOrDefault(c, 0) + 1);*

*}*

*int count = 0;*  *for (char c : str2.toCharArray()) {*  *if (charCountMap.containsKey(c)) {*

*count++;*

*charCountMap.put(c, charCountMap.get(c) - 1);*  *if (charCountMap.get(c) == 0) {*

*charCountMap.remove(c);*

*}*

*}*

*}*

*return count;*

*}*

*}*

***B)* Count number of free cell present in the Matrix**

Given a Matrix size **N\*N** and an integer **K**. Initially, the matrix contains only **0**. You are given K tasks and for each task, you are given two coordinates (**r, c**). Where coordinates (r, c) denotes the **rth** row and the **cth** column of the given matrix. You have to perform each task sequentially in the given order. For each task, You have to put 1 in all rth row cells and all the cth column cells. After each task, You have to calculate the number of 0 present in the matrix and return it.

**Examples:**

***Input****: N = 3, K = 3*

*1 1*

1. *2*
2. *1*

***Output****: 4 2 1*

***Explanation:*** *After 1st Operation, all the cells of the 2nd row and 2nd column will be filled by 1. So, the remaining cell with the value 0 will be 4. After 2nd operation, all the second row and all the third columns will be filled by 1. So, the remaining cell with value will be 2. After 3rd operation number of cells having the value 0 will be 1.*

***Input****: N = 2, K = 2*

*0 1*

*0 0*

***Output****: 1 0*

***Explanation:*** *After 1st operation, all the cells of the 1st row and 2nd column will be filled by 1. So, the remaining cell with the value 0 will be 1. After 2nd operation, all the cells of the 1st row and 1st column will be filled by 1. So, the remaining cell with the value 0 will be 0.*

*import java.util.Arrays;*

*import java.util.Scanner;* *public class MatrixTasks {*  *public static void main(String[] args) {*

*Scanner scanner = new Scanner(System.in);*  *System.out.print("Enter the size of the matrix: ");*  *int N = scanner.nextInt();*

*System.out.print("Enter the number of tasks: ");*  *int K = scanner.nextInt();*  *int[][] matrix = new int[N][N];*  *int[] tasksR = new int[K];*  *int[] tasksC = new int[K];*

*System.out.println("Enter the coordinates of each task (r, c):");*  *for (int i = 0; i < K; i++) {*  *tasksR[i] = scanner.nextInt();*  *tasksC[i] = scanner.nextInt();*

*}*

*for (int i = 0; i < K; i++) {*  *int r = tasksR[i];*  *int c = tasksC[i];*  *for (int j = 0; j < N; j++) {*

*matrix[r - 1][j] = 1; // Set all cells in rth row to 1*  *matrix[j][c - 1] = 1; // Set all cells in cth column to 1*

*}*

*int zerosCount = countZeros(matrix);*

*System.out.print(zerosCount + " ");*

*}*

*scanner.close();*

*}*

*private static int countZeros(int[][] matrix) {*

*int count = 0;*  *for (int[] row : matrix) {*  *for (int cell : row) {*  *if (cell == 0) {*  *count++;*

*}*

*}*

*}*

*return count;*

*}*

*}*

***33)A)* Modify a matrix by replacing each element with the maximum of its left or right diagonal sum**

Given a matrix **mat[][]** with dimensions **M \* N**, the task is to replace each matrix elements with the maximum sum of its left or right diagonal.

**Examples:**

***Input:*** *mat[][] = {{5, 2, 1}, {7, 2, 6}, {3, 1, 9}}* ***Output:***

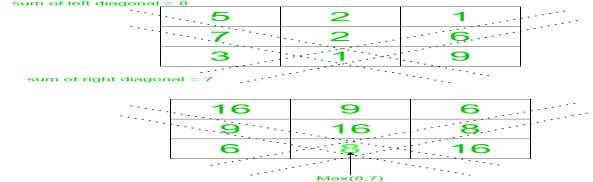
*16 9 6*

*9 16 8*

*6 8 16*

***Explanation:***

*Replace each element with max(sum of right diagonal, sum of left diagonal). Follow the diagram below to understand more clearly.*



|  |  |  |
| --- | --- | --- |
| ***nput:*** *mat[][] = {{1, 2}, {3, 4}}* | | |
| ***Output:*** | |  |
| *5 5*  *5 5* |  |

*import java.util.Scanner; public class DiagonalMaxSum {*  *public static void main(String[] args) {*  *Scanner scanner = new Scanner(System.in);*  *System.out.print("Enter the number of rows (M): ");*  *int M = scanner.nextInt();*

*System.out.print("Enter the number of columns (N): ");*  *int N = scanner.nextInt();*  *int[][] mat = new int[M][N];*

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

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

*mat[i][j] = scanner.nextInt();*

*}*

*}*

*int[][] result = new int[M][N];*  *for (int i = 0; i < M; i++) {*  *for (int j = 0; j < N; j++) {*  *int leftSum = 0;*  *int rightSum = 0;*

*for (int k = 0; k < Math.min(i, j); k++) {*  *leftSum += mat[i - k][j - k];*

*}*

*for (int k = 0; k < Math.min(i, N - j - 1); k++) {*  *rightSum += mat[i - k][j + k];*

*}*

*result[i][j] = Math.max(leftSum, rightSum);*

*}*

*}*

*System.out.println("Result:");*  *for (int i = 0; i < M; i++) {*  *for (int j = 0; j < N; j++) {*

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

*}*

*System.out.println();*

*}*

*scanner.close();*

*}*

*}*

***B)* Print characters in decreasing order of frequency**

Given string **str**, the task is to print the characters in decreasing order of their frequency. If the frequency of two characters is the same then sort them in descending order alphabetically. **Examples:**

***Input:*** *str = “geeksforgeeks”* ***Output:***

*e – 4 s – 2 k – 2 g – 2*

*r – 1 o – 1 f – 1*

***Input:*** *str = “bbcc”* ***Output:*** *c – 2 b – 2*

import java.util.HashMap;import java.util.Map; import java.util.PriorityQueue; import java.util.Scanner; public class CharacterFrequency { public static void main(String[] args) { Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: ");

String str = scanner.nextLine();

Map<Character, Integer> frequencyMap = new HashMap<>(); for (char c : str.toCharArray()) {

frequencyMap.put(c, frequencyMap.getOrDefault(c, 0) + 1);

}

PriorityQueue<Map.Entry<Character, Integer>> maxHeap = new PriorityQueue<>((a, b) -> { if (a.getValue().equals(b.getValue())) {

return b.getKey() - a.getKey();

} else {

return b.getValue() - a.getValue();

}

});

maxHeap.addAll(frequencyMap.entrySet());

while (!maxHeap.isEmpty()) {

Map.Entry<Character, Integer> entry = maxHeap.poll();

System.out.println(entry.getKey() + " - " + entry.getValue());

}

scanner.close();

} }

***34) A)* Count of unique rows in a given Matrix**

Given a **2D** matrix arr of size **N\*M** containing lowercase English letters, the task is to find the number of unique rows in the given matrix. **Examples:**

***Input:*** *arr[][]= { {‘a’, ‘b’, ‘c’, ‘d’},*

*{‘a’, ‘e’, ‘f’, ‘r’},*

*{‘a’, ‘b’, ‘c’, ‘d’},*

*{‘z’, ‘c’, ‘e’, ‘f’} }*

***Output:*** *2*

***Explanation:*** *The 2nd and the 4th row are unique.*

***Input:*** *arr[][]={{‘a’, ‘c’},*

*{‘b’, ‘d’},*

*{‘e’, ‘f’}}*

***Output:*** *3*

*import java.util.\*; public class Main {*

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

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

*System.out.print("Enter the number of rows in the matrix: ");* *int n = scanner.nextInt();*

*System.out.print("Enter the number of columns in the matrix: ");* *int m = scanner.nextInt();* *char[][] arr = new char[n][m];*

*System.out.println("Enter the elements of the matrix:");* *for (int i = 0; i < n; i++) {* *for (int j = 0; j < m; j++) {* *arr[i][j] = scanner.next().charAt(0);*

*}*

*}*

*int uniqueRowsCount = countUniqueRows(arr);*

*System.out.println("Number of unique rows: " + uniqueRowsCount);* *scanner.close();*

*}*

*public static int countUniqueRows(char[][] arr) {* *Set<String> uniqueRows = new HashSet<>();*

*for (char[] row : arr) {*

*String rowString = new String(row);*

*uniqueRows.add(rowString);*

*}*

*return uniqueRows.size();*

*}*

*}*

***B)* Count of strings with frequency of each character at most X and length at least Y**

Given an array **arr[]** of strings and integers **X** and **Y**, the task is to find the count of strings with **frequency** of each character **at most X** and **length** of the string **at least Y**. **Examples:**

***Input:*** *arr[] = { “ab”, “derdee”, “erre” }, X = 2, Y = 4*

***Output:*** *1*

***Explanation:*** *Strings with character frequency at most 2 and length at least 4 is “erre”. Hence count is 1*

***Input:*** *arr[] = {“ag”, “ka”, “nanana”}, X = 3, Y = 2* ***Output:*** *3*

import java.util.HashMap; import java.util.Map; import java.util.Scanner;

public class CountStrings {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of strings: "); int n = scanner.nextInt(); scanner.nextLine();

String[] strings = new String[n];

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

for (int i = 0; i < n; i++) { strings[i] = scanner.nextLine();

}

System.out.print("Enter the maximum frequency of each character (X): "); int maxFrequency = scanner.nextInt();

System.out.print("Enter the minimum length of the string (Y): "); int minLength = scanner.nextInt();

int count = countStrings(strings, maxFrequency, minLength);

System.out.println("Count of strings with frequency of each character at most " + maxFrequency + " and length at least " + minLength + ": " + count);

}

private static int countStrings(String[] strings, int maxFrequency, int minLength) { int count = 0;

for (String str : strings) {

if (hasValidConditions(str, maxFrequency, minLength)) { count++;

}

}

return count;

}

private static boolean hasValidConditions(String str, int maxFrequency, int minLength) { Map<Character, Integer> frequencyMap = new HashMap<>(); for (char ch : str.toCharArray()) {

frequencyMap.put(ch, frequencyMap.getOrDefault(ch, 0) + 1);

}

for (int frequency : frequencyMap.values()) { if (frequency > maxFrequency) { return false;

}

}

return str.length() >= minLength;

}

}

***35) A)* Sum of all elements repeating ‘k’ times in an array**

Given an array, we have to find the sum of all the elements repeating k times in an array. We need to consider every repeating element just once in the

sum.

**Examples:**

**Input :** arr[] = {2, 3, 9, 9}

k = 1

## Output : 5

2 + 3 = 5

**Input :** arr[] = {9, 8, 8, 8, 10, 4}

k = 3

## Output : 8

import java.util.HashMap; import java.util.Scanner; public class Main { 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[] arr = new int[n];

System.out.println("Enter the elements of the array:"); for (int i = 0; i < n; i++) { arr[i] = scanner.nextInt();

}

System.out.print("Enter the value of k: "); int k = scanner.nextInt();

scanner.close();

HashMap<Integer, Integer> frequencyMap = new HashMap<>(); for (int i : arr) {

frequencyMap.put(i, frequencyMap.getOrDefault(i, 0) + 1);

} int sum = 0; for (int i : frequencyMap.keySet()) { if (frequencyMap.get(i) == k) { sum += i;

}

}

System.out.println("Sum: " + sum);

}

}

***B)* Most frequent word in first String which is not present in second String**

Given two string ‘S1’ and ‘S2’, the task is to return the most frequent (which is used the maximum number of times) word from ‘S1’ that is not present in ‘S2’. If more than one word is possible then print lexicographically smallest among them.

**Examples:**

***Input:*** *S1 = “geeks for geeks is best place to learn”, S2 = “bad place”* ***Output:*** *geeks*

*“geeks” is the most frequent word in S1 and is also not present in S2.*

*The frequency of “geeks” is 2*

***Input:*** *S1 = “the quick brown fox jumps over the lazy dog”, S2 = “the brown fox jumps”*

***Output:*** *dog*

*All the words have frequency 1.*

*The lexicographically smallest word is “dog”*

*import java.util.\*; public class Main {*

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

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

*System.out.print("Enter string S1: ");*

*String s1 = scanner.nextLine();*

*System.out.print("Enter string S2: ");*

*String s2 = scanner.nextLine();*

*String result = findMostFrequentWord(s1, s2);* *System.out.println("Most frequent word: " + result);* *scanner.close();*

*}*

*public static String findMostFrequentWord(String s1, String s2) {*

*Map<String, Integer> frequencyMap = new HashMap<>();* *String[] words1 = s1.split(" ");* *for (String word : words1) {*

*frequencyMap.put(word, frequencyMap.getOrDefault(word, 0) + 1);*

*}*

*String[] words2 = s2.split(" ");* *for (String word : words2) {* *frequencyMap.remove(word);*

*}*

*int maxFrequency = 0;* *String mostFrequentWord = "";*

*for (Map.Entry<String, Integer> entry : frequencyMap.entrySet()) {* *String word = entry.getKey();* *int frequency = entry.getValue();*

*if (frequency > maxFrequency || (frequency == maxFrequency && word.compareTo(mostFrequentWord) < 0)) {* *maxFrequency = frequency;* *mostFrequentWord = word;*

*}*

*}*

*return mostFrequentWord;*

*}*

*}*

***36) A)* Find element with highest frequency in given nested Array**

Given an [array](https://www.geeksforgeeks.org/introduction-to-arrays/) **arr[]** of **N** integers. The task is to create a frequency array **freq[]** of the given array **arr[]** and find the maximum element of the frequency array. If two elements have the same frequency in the array **freq[]**, then return the element which has a smaller value.

**Examples:**

***Input:*** *arr[] = {1, 1, 1, 2, 3, 2, 2, 3, 5, 5, 5, 5, 4, 4, 4, 4, 4};*

***Output:*** *3* ***Explanation:*** *frequency of elements is given by: val -> freq[]*

1. *-> 3*
2. *-> 3*
3. *-> 2*
4. *-> 5*
5. *-> 4*

*Element 3 has the maximum frequency in the frequency array.*

***Input:*** *arr[] = { 3, 5, 15, 51, 15, 14, 14, 14, 14, 4};* ***Output:*** *1* ***Explanation:*** *frequency of elements is given by: val -> freq[]*

1. *-> 1*
2. *-> 1*
3. *-> 1*
4. *-> 4*
5. *-> 2*

*51 -> 1*

*Element 1 has the maximum frequency in the frequency array.*

*import java.util.HashMap; import java.util.Map;* *import java.util.Scanner;* *public class Main {*  *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[] arr = new int[n];*

*System.out.print("Enter the elements in the array: ");*

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

*arr[i] = scanner.nextInt();*

*}*

*scanner.close();*

*Map<Integer, Integer> frequencyMap = new HashMap<>();*  *for (int num : arr) {*

*frequencyMap.put(num, frequencyMap.getOrDefault(num, 0) + 1);*

*}*

*int maxFrequency = 0;*

*int maxFrequencyElement = Integer.MAX\_VALUE;*  *for (Map.Entry<Integer, Integer> entry : frequencyMap.entrySet()) {*  *int element = entry.getKey();*  *int frequency = entry.getValue();*

*if (frequency > maxFrequency || (frequency == maxFrequency && element <*

*maxFrequencyElement)) {*  *maxFrequency = frequency;*

*maxFrequencyElement = element;*

*}*

*}*

*System.out.println("Max frequency element: " + maxFrequencyElement);*

*}*

*}*

***B)* Remove duplicates from string keeping the order according to last occurrences**

Given a string, remove duplicate characters from the string, retaining the last occurrence of the duplicate characters. Assume the characters are case-sensitive.

**Examples:**

***Input :*** *geeksforgeeks*

***Output :*** *forgeks*

*Explanation : Please note that we keep only last occurrences of repeating characters in same order as they appear in input. If we see result from right side, we can notice that we keep last ‘s’, then last ‘k’ , and so on.*

***Input :*** *hi this is sample test*

***Output :*** *hiampl est*

*Explanation : Here, the output contains last occurrence of every character, even ” “(spaces), and removing the duplicates. Like in this example, there are 4 spaces count, so we have only the last occurrence of space in it removing the others. And there is only last occurrence of each character without repetition.*

*import java.util.LinkedHashMap; import java.util.Map;* *import java.util.Scanner;* *public class Main {*

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

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

*System.out.print("Enter the string: ");*  *String input = scanner.nextLine();*  *scanner.close();*

*Map<Character, Integer> charMap = new LinkedHashMap<>();*

*for (int i = input.length() - 1; i >= 0; i--) {*  *char c = input.charAt(i);* *charMap.put(c, i);*

*}*

*StringBuilder output = new StringBuilder();*  *for (char c : charMap.keySet()) {*  *output.append(c);*

*}*

*System.out.println("Output: " + output.toString());*

*}*

*}*

***37) A)* Find element in a sorted array whose frequency is greater than or equal to n/2.**

Given a sorted array of length n, find the number in array that appears more than or equal to n/2 times. It is given that such element always exists.

**Examples:**

Input : 2 3 3 4

Output : 3

Input : 3 4 5 5 5

Output : 5

Input : 1 1 1 2 3

Output : 1

*import java.util.Arrays; import java.util.Scanner;* *public class MajorityElement {*

*public static int findMajorityElement(int[] arr) {*  *int count = 1;*

*int majorityElement = arr[0];*

*for (int i = 1; i < arr.length; i++) {*  *if (arr[i] == majorityElement) {*

*count++;*  *} else {*  *count--;*  *if (count == 0) {*  *majorityElement = arr[i];*

*count = 1;*

*}*

*}*

*}*

*return majorityElement;*

*}*

*public static void main(String[] args) {*  *Scanner scanner = new Scanner(System.in);*  *System.out.print("Enter the length of the array: ");*  *int length = scanner.nextInt();*  *int[] arr = new int[length];*

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

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

*arr[i] = scanner.nextInt();*

*}*

*int majorityElement = findMajorityElement(arr);*

*System.out.println("Majority element: " + majorityElement);*

*}*

*}*

***B)* Remove even frequency characters from the string**

Given a string ‘str’, the task is to remove all the characters from the string that have even frequencies.

**Examples:**

**Input:** str = "aabbbddeeecc"

## Output: bbbeee

The characters a, d, c have even frequencies So, they are removed from the string.

**Input:** str = "zzzxxweeerr"

**Output:** zzzweee

*import java.util.HashMap; import java.util.Map;* *import java.util.Scanner;*

*public class RemoveEvenFrequencyCharacters {*  *public static String removeCharacters(String str) {*

*Map<Character, Integer> frequencyMap = new HashMap<>();*  *for (char ch : str.toCharArray()) {*

*frequencyMap.put(ch, frequencyMap.getOrDefault(ch, 0) + 1);*

*}*

*StringBuilder result = new StringBuilder();*  *for (char ch : str.toCharArray()) {*  *if (frequencyMap.get(ch) % 2 != 0) {*

*result.append(ch);*

*}*

*}*

*return result.toString();*

*}*

*public static void main(String[] args) {*  *Scanner scanner = new Scanner(System.in);*

*System.out.print("Enter a string: ");*

*String str = scanner.nextLine();*

*String result = removeCharacters(str);*  *System.out.println("Output: " + result);*

*scanner.close();*

*}*

*}*