JAVA TASK

Find the Missing Number

Problem Statement:
Given an array arr[] of size N-1 with integers in the range of [1, N], the task is to find the missing number from the first N integers.
Note: There are no duplicates in the list.

Input Format
First Line : The array size : N
Second Line : The array elements

Output Format
Print the missing number .
SAMPLE INPUT 1
8
1 2 4 6 3 7 8
SAMPLE OUTPUT 1
5
Input: arr[] = {1, 2, 4, 6, 3, 7, 8} , N = 8

Output: 5
Explanation: Here the size of the array is 8, so the range will be [1, 8]
. The missing number between 1 to 8 is 5
Input: arr[] = {1, 2, 3, 5}, N = 5
Output: 4
Explanation: Here the size of the array is 4, so the range will be [1, 5].
The missing number between 1 to 5 is 4

Code :

```
import java.util.Scanner;

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

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

```
        for(int i = 0; i < n-1; i++) {
            arr[i] = sc.nextInt();
        }
        sc.close();

        int sum = 0;

        for(int i = 0; i < n-1; i++) {
            sum += arr[i];
        }

        int Missing = ((n*(n+1))/2) - sum;

        System.out.println("Missing number : " + Missing);

    }
}
```

```
D:\230701298>javac missingNum.java

D:\230701298>java missingNum
5
1 2 3 4
Missing number : 5

D:\230701298>java missingNum
5
1 3 4 5
Missing number : 2
```

Move all Zeros to the End of the Array

Problem Statement:

Given an array of N elements, Your task is to move the Zeroes to the end of the Array.

Input Format
First Line : Array Size : N
Second Line: Array elements separated by space
Output Format
First line: Array elements arranged as zeroes at the end.

SAMPLE INPUT
5
1 0 2 0 3 0 4 0

SAMPLE OUTPUT
1 2 3 4 0 0 0

Code :

```
import java.util.Scanner;

public class moveZerosToEnd {

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

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

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

        int a = 0;

        for(int i = 0; i < n; i++) {
            if(arr[i] != 0) {
                int temp = arr[i];
                arr[i] = arr[a];
                arr[a] = temp;
                a++;
            }
        }
        for(int i = 0; i < n; i++)  System.out.print(arr[i] + "
");
    }
}
```

```
D:\230701298>javac moveZerosToEnd.java

D:\230701298>java moveZerosToEnd
5
1 2 0 3 0
1 2 3 0 0
D:\230701298>java moveZerosToEnd
5
0 10 5 0 2
10 5 2 0 0
```

String Buffer with Roman Numerals

Problem Statement:
Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.
Symbol Value
I 1
V 5
X 10
L 50
C 100
D 500
M 1000

For example, 2 is written as II in Roman numerals, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.
Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:
• I can be placed before V (5) and X (10) to make 4 and 9.
• X can be placed before L (50) and C (100) to make 40 and 90.
• C can be placed before D (500) and M (1000) to make 400 and 900.
Constraints:
• 1 <= s.length<= 15
• s contains only the characters (‘I’, ‘V’, ‘X’, ‘L’, ‘C’, ‘D’, ‘M’) .
• It is guaranteed that s is a valid roman numeral in the range [1, 3999].

Input Format:
Read the string

Output Format:
Print the numeral equivalent to roman.

SAMPLE INPUT
Input: s = “III”
Input: s = “LVIII”
Input: s = “MCMXCIV”

SAMPLE OUTPUT
Output: 3 // Explanation: III = 3.
Output: 58 // Explanation: L = 50, V= 5, III = 3
Output: 1994 //Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.

Code :

```
import java.util.Scanner;

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

        String s = sc.nextLine();
        sc.close();

        int tot = 0;
        int p = 0;

        for(int i = 0; i < s.length(); i++) {
            char l = s.charAt(i);
            int val = 0;

            switch(l) {
                case 'I' :
                    val = 1;
                    break;
                case 'V' :
                    val = 5;
                    break;
                case 'X' :
                    val = 10;
```

```
                    break;
                case 'L' :
                    val = 50;
                    break;
                case 'C' :
                    val = 100;
                    break;
                case 'D' :
                    val = 500;
                    break;
                case 'M' :
                    val = 1000;
                    break;
            }
            if(val > p) tot += val - 2 * p;
            else    tot += val;

            p = val;
        }
        System.out.println(tot);
    }
}
```

```
D:\230701298>javac Roman.java

D:\230701298>java Roman
III
3

D:\230701298>java Roman
LVIII
58

D:\230701298>java Roman
MCMXCIV
1994
```

Palindrome

Problem Statement:

A phrase is a palindrome if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and

backward. Alphanumeric characters include letters and numbers.

Given a string s, return true if it is a palindrome, or false otherwise.

Input Format:
Read the sentence

Output Format:
Print true or false

SAMPLE INPUT

Input: s = “A man, a plan, a canal: Panama”

SAMPLE OUTPUT
True

// Explanation: “amanaplanacanalpanama” is a palindrome

Code :

```
import java.util.Scanner;

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

        String sentence= sc.nextLine();
        sc.close();

        sentence =
sentence.toLowerCase().replaceAll("[^a-zA-Z0-9]", "");
        int l = sentence.length() - 1;

        boolean pal = true;
        for(int i = 0; i < l/2; i++) {
            if(sentence.charAt(i) != sentence.charAt(l)) {
                pal = false;
                break;
            }
            l--;
```

```
        }

        if(pal) System.out.println("True");

        else    System.out.println("False");
    }
}
```

```
D:\230701298>javac IsPalindrome.java

D:\230701298>java IsPalindrome
A man, a plan, a canal : Panama
True

D:\230701298>java IsPalindrome
A man, a plan.
False
```