Rajalakshmi Engineering College

Name: Adithya B
Email: 240701018@rajalakshmi.edu.in
Roll no: 240701018
Phone: 9444117405
Branch: REC
Department: CSE - Section 10
Batch: 2028
Degree: B.E - CSE

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 3_PAH

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Priya is building a system to automate image transformations using matrix operations. To do this, she needs to multiply two matrices representing pixel data and transformation rules.

Help Priya perform matrix multiplication and print the resulting matrix if the operation is valid.

Input Format

The first line of input consists of two int values, representing the number of rows R1 and columns C1 of the first matrix.

The next R1 × C1 integers represent the elements of the first matrix.

The next line consists of two int values, representing the number of rows R2 and columns C2 of the second matrix.

The next R2 × C2 integers represent the elements of the second matrix.

Output Format

If matrix multiplication is possible, print R1 lines, each containing C2 space-separated int values representing the resulting matrix.

Otherwise, print "Matrix multiplication not possible".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2 3
1 2 3
4 5 6
3 2
7 8
9 10
11 12

Output: 58 64
139 154

Answer

```
import java.util.Scanner;

class MatrixOperations {

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

        int R1 = scanner.nextInt();
        int C1 = scanner.nextInt();
        int[][] matrix1 = new int[R1][C1];

        for (int i = 0; i < R1; i++) {
            for (int j = 0; j < C1; j++) {
                matrix1[i][j] = scanner.nextInt();
            }
        }

        int R2 = scanner.nextInt();
        int C2 = scanner.nextInt();
        int[][] matrix2 = new int[R2][C2];

        for (int i = 0; i < R2; i++) {
            for (int j = 0; j < C2; j++) {
                matrix2[i][j] = scanner.nextInt();
            }
        }

        if (C1 != R2) {
            System.out.println("Matrix multiplication not possible");
            return;
        }

        int[][] resultMatrix = new int[R1][C2];
        for (int i = 0; i < R1; i++) {
            for (int j = 0; j < C2; j++) {
                for (int k = 0; k < C1; k++) {
                    resultMatrix[i][j] += matrix1[i][k] * matrix2[k][j];
                }
            }
        }

        for (int i = 0; i < R1; i++) {
            for (int j = 0; j < C2; j++) {
                System.out.print(resultMatrix[i][j] + " ");
            }
            System.out.println();
        }
        scanner.close();
    }
}
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Egath is participating in a coding hackathon, and one of the challenges requires him to work with an array of integers. The task is to remove exactly one element from the array such that the sum of the remaining elements is a prime number.

Help Egath find the first possible prime sum by removing one element or determining if no such prime sum can be achieved.

Input Format

The first line of input consists of an integer N, representing the number of elements in the array.

The second line consists of N space-separated integers, representing the array elements.

Output Format

If removing one element results in a prime sum, print the sum.

If no such prime sum can be achieved by removing exactly one element, print "No valid prime sum found".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1 2 3

Output: 5

Answer

```
import java.util.Scanner;

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

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

        boolean foundPrimeSum = false;
        for (int i = 0; i < n && !foundPrimeSum; i++) {
            int remainingSum = totalSum - nums[i];
            if (remainingSum > 1) {
                boolean isPrime = true;

                if (remainingSum > 3) {
                    if (remainingSum % 2 == 0 || remainingSum % 3 == 0) {
                        isPrime = false;
                    } else {
                        for (int j = 5; j * j <= remainingSum; j += 6) {
                            if (remainingSum % j == 0 || remainingSum % (j + 2) == 0) {
                                isPrime = false;
                                break;
                            }
                        }
                    }
                }

                if (isPrime) {
                    System.out.println(remainingSum);
                    foundPrimeSum = true;
                }
            }
        }

        if (!foundPrimeSum) {
            System.out.println("No valid prime sum found");
        }
        scanner.close();
    }
}
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

In a customer loyalty program, reward points are logged in a sorted array as customers make transactions. Occasionally, due to system errors, duplicate entries for the same transaction may appear. To ensure accurate reward calculations, it's crucial to remove these duplicates from the list.

Write a program to process the array of reward points, removing any duplicates while preserving the order of unique entries. The program should then display the cleaned list of unique reward points and the total count of these unique points.

Input Format

The first line of input consists of an integer N, representing the number of reward points.

The second line consists of N space-separated integers, representing the reward points in sorted order.

Output Format

The first line of output prints the cleaned list of unique reward points separated by a space.

The second line of output prints an integer representing the total count of unique reward points.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
100 100 200

Output: 100 200
2

Answer

```
import java.util.Scanner;

class RemoveDuplicates {
  public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] nums = new int[n];
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }
        if (nums.length == 0) {
            System.out.println();
            System.out.println(0);
            scanner.close();
            return;
        }
        int uniqueIndex = 1;
        for (int i = 1; i < nums.length; i++) {
            if (nums[i] != nums[i - 1]) {
                nums[uniqueIndex] = nums[i];
                uniqueIndex++;
            }
        }
        for (int i = 0; i < uniqueIndex; i++) {
            System.out.print(nums[i] + " ");
        }
        System.out.println();
        System.out.println(uniqueIndex);

        scanner.close();
    }
}
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

Eminem is a billiard player who enjoys playing billiards and also likes solving mathematical puzzles. He notices that the billiard balls on the table are arranged in a grid, and he is curious to find the sum of the numbers written on each ball.

Write a program to find the sum of all the numbers written on each ball in the grid.

Input Format

The first line of input consists of an integer N, representing the number of rows.

The second line consists of an integer M, representing the number of columns.

The following lines N lines consist of M space-separated integers, representing the numbers written on each ball.

Output Format

The output prints an integer representing the sum of all the numbers written on each ball.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
3
1 2 3
4 5 6
7 8 9
Output: 45

Answer

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int M = sc.nextInt();

        int sum = 0;
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++) {
                sum += sc.nextInt();
            }
        }
        System.out.println(sum);
    }
}
```

Status : Correct *Marks : 10/10*