

# Rajalakshmi Engineering College

Name: Gnaana Kumaar

Email: 241001062@rajalakshmi.edu.in

Roll no: 241001062

Phone: 9790249960

Branch: REC

Department: IT - Section 4

Batch: 2028

Degree: B.E - IT

Scan to verify results



## 2024\_28\_III\_OOPS Using Java Lab

### **REC\_2028\_OOPS using Java\_Week 3\_CY**

Attempt : 1

Total Mark : 40

Marks Obtained : 40

### **Section 1 : Coding**

#### **1. Problem Statement**

Nikila is working as an intern in a software firm and is practicing with a matrix where each row represents a set of numerical values. Her task is to identify the row with the highest sum of its elements and remove that row from the matrix. After removing the row with the highest sum, Nikila needs to print the updated matrix.

Your task is to help Nikila in implementing the same. If there are two or more rows that have same the highest sum, the firstly encountered row is deleted.

#### ***Input Format***

The first line of the input consists of two space-separated integers, R and C, representing the number of rows and columns in the matrix, respectively.

The following R lines each contain, C space-separated integers representing the matrix elements.

### ***Output Format***

The output prints the matrix after removing the row with the highest sum. Each row should be printed on a new line, with elements separated by a space.

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: 2 2

1 2

3 4

Output: 1 2

### ***Answer***

```
// You are using Java
import java.util.Scanner;

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

        // Read dimensions
        int R = sc.nextInt();
        int C = sc.nextInt();

        int[][] matrix = new int[R][C];

        // Read matrix elements
        for (int i = 0; i < R; i++) {
            for (int j = 0; j < C; j++) {
                matrix[i][j] = sc.nextInt();
            }
        }

        // Find the row with the maximum sum
        int maxSum = Integer.MIN_VALUE;
```

```

int rowToRemove = -1;

for (int i = 0; i < R; i++) {
    int sum = 0;
    for (int j = 0; j < C; j++) {
        sum += matrix[i][j];
    }
    if (sum > maxSum) {
        maxSum = sum;
        rowToRemove = i;
    }
}

// Print the matrix excluding the row with the max sum
for (int i = 0; i < R; i++) {
    if (i == rowToRemove) continue; // skip the row with max sum
    for (int j = 0; j < C; j++) {
        System.out.print(matrix[i][j]);
        if (j != C - 1) System.out.print(" ");
    }
    System.out.println();
}

sc.close();
}
}

```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement:

Imagine you have an array of integer values, and you're tasked with identifying a pair of elements within the array. This pair of elements should have a sum that is the closest to zero when compared to any other pair in the array.

Your goal is to create a program that solves this problem efficiently. The program should accept an array of integers and return the pair of elements whose sum is closest to zero.

### ***Input Format***

The first line of the input is an integer N representing the size of the array.

The second line of the input contains N space-separated integer values.

### ***Output Format***

The output is displayed in the following format:

"Pair with the sum closest to zero: {value} and {value}"

Refer to the sample output for formatting specifications.

### ***Sample Test Case***

Input: 5

9 10 -3 -5 -2

Output: Pair with the sum closest to zero: 9 and -5

### ***Answer***

```
// You are using Java
import java.util.Scanner;

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

        // Read array size
        int N = sc.nextInt();
        int[] arr = new int[N];

        // Read array elements
        for (int i = 0; i < N; i++) {
            arr[i] = sc.nextInt();
        }

        // Initialize variables to track closest pair
        int closestSum = Integer.MAX_VALUE;
        int firstElement = 0, secondElement = 0;
```

```

        // Check all pairs to find the closest sum to zero
        for (int i = 0; i < N - 1; i++) {
            for (int j = i + 1; j < N; j++) {
                int sum = arr[i] + arr[j];
                if (Math.abs(sum) < Math.abs(closestSum)) {
                    closestSum = sum;
                    firstElement = arr[i];
                    secondElement = arr[j];
                }
            }
        }

        // Print the result
        System.out.println("Pair with the sum closest to zero: " + firstElement +
and " + secondElement);

        sc.close();
    }
}

```

**Status : Correct**

**Marks : 10/10**

### 3. Problem Statement

Rina is managing the inventory for a library, where each row of a 2D matrix represents the number of different genres of books available on each shelf.

She wants to perform the following operations:

Transformation: Replace each element in a row with the sum of all elements in that row.  
Merging: After transformation, Rina will provide one additional matrix, and specify whether to merge the transformed matrix with this new matrix row-wise or column-wise.

#### ***Input Format***

The first line contains two integers R and C, representing the number of rows and columns of the initial matrix.

The next R lines contain C space-separated integers, representing the book

counts in the library.

The next line contains two integers MR and MC, representing the dimensions of the second matrix (to be merged).

The next MR lines contain MC space-separated integers, representing the second matrix.

The last line contains an integer mergeType:

- 0 Row-wise merging (append the second matrix below the transformed matrix).
- 1 Column-wise merging (append the second matrix to the right of the transformed matrix).

#### ***Output Format***

The output prints "Transformed matrix: " followed by the transformed 2D matrix where each element in a row is replaced with the sum of the elements in that row.

The output prints "Final merged matrix: ", followed by the merging based on mergeType.

Refer to the sample output for formatting specifications.

#### ***Sample Test Case***

Input: 3 4  
8 2 4 9  
4 5 6 1  
7 8 9 3  
2 4  
3 5 7 2  
6 1 4 9  
0

Output: Transformed matrix:

23 23 23 23  
16 16 16 16  
27 27 27 27

Final merged matrix:

23 23 23 23  
16 16 16 16  
27 27 27 27  
3 5 7 2  
6 1 4 9

### Answer

```
// You are using Java
import java.util.Scanner;

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

        // Read first matrix dimensions
        int R = sc.nextInt();
        int C = sc.nextInt();

        int[][] matrix1 = new int[R][C];

        // Read first matrix elements
        for (int i = 0; i < R; i++) {
            for (int j = 0; j < C; j++) {
                matrix1[i][j] = sc.nextInt();
            }
        }

        // Transform the first matrix: replace each element with the row sum
        int[][] transformed = new int[R][C];
        for (int i = 0; i < R; i++) {
            int rowSum = 0;
            for (int j = 0; j < C; j++) {
                rowSum += matrix1[i][j];
            }
            for (int j = 0; j < C; j++) {
                transformed[i][j] = rowSum;
            }
        }

        // Read second matrix dimensions
        int MR = sc.nextInt();
        int MC = sc.nextInt();
```

```
int[][] matrix2 = new int[MR][MC];

// Read second matrix elements
for (int i = 0; i < MR; i++) {
    for (int j = 0; j < MC; j++) {
        matrix2[i][j] = sc.nextInt();
    }
}

// Read merge type
int mergeType = sc.nextInt();

// Print transformed matrix
System.out.print("Transformed matrix: ");
for (int i = 0; i < R; i++) {
    for (int j = 0; j < C; j++) {
        System.out.print(transformed[i][j]);
        if (i != R - 1 || j != C - 1) System.out.print(" ");
    }
}
System.out.println();

// Merge matrices
if (mergeType == 0) {
    // Row-wise merge (append second matrix below)
    // Dimensions of final matrix: (R + MR) x max(C, MC)
    int finalRows = R + MR;
    int finalCols = Math.max(C, MC);
    int[][] merged = new int[finalRows][finalCols];

    // Copy transformed matrix
    for (int i = 0; i < R; i++) {
        for (int j = 0; j < C; j++) {
            merged[i][j] = transformed[i][j];
        }
    }

    // Copy second matrix
    for (int i = 0; i < MR; i++) {
        for (int j = 0; j < MC; j++) {
            merged[R + i][j] = matrix2[i][j];
        }
    }
}
```

```
        }
    }

    // Print final merged matrix
    System.out.print("Final merged matrix: ");
    for (int i = 0; i < finalRows; i++) {
        for (int j = 0; j < finalCols; j++) {
            System.out.print(merged[i][j]);
            if (i != finalRows - 1 || j != finalCols - 1) System.out.print(" ");
        }
    }
    System.out.println();

} else if (mergeType == 1) {
    // Column-wise merge (append second matrix to the right)
    // Dimensions of final matrix: max(R, MR) x (C + MC)
    int finalRows = Math.max(R, MR);
    int finalCols = C + MC;
    int[][] merged = new int[finalRows][finalCols];

    // Initialize merged matrix with 0 (default in Java)

    // Copy transformed matrix to left part
    for (int i = 0; i < R; i++) {
        for (int j = 0; j < C; j++) {
            merged[i][j] = transformed[i][j];
        }
    }

    // Copy second matrix to right part
    for (int i = 0; i < MR; i++) {
        for (int j = 0; j < MC; j++) {
            merged[i][C + j] = matrix2[i][j];
        }
    }

    // Print final merged matrix
    System.out.print("Final merged matrix: ");
    for (int i = 0; i < finalRows; i++) {
        for (int j = 0; j < finalCols; j++) {
            System.out.print(merged[i][j]);
            if (i != finalRows - 1 || j != finalCols - 1) System.out.print(" ");
        }
    }
}
```

```
        }
    }
    System.out.println();
}
sc.close();
}
}
```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement:

Mason is participating in a coding challenge where he must manipulate an integer array. His task is to replace every element in the array with the next greatest element to its right. The last element of the array remains unchanged, as there is no element to its right.

Your job is to help Mason write a program that performs this transformation and outputs the modified array.

#### *Input Format*

The first line of input contains an integer  $n$  representing the number of elements in the array.

The second line of input contains  $n$  space-separated integers representing the elements of the array.

#### *Output Format*

The output prints the modified array of  $n$  integers, where each element (except the last one) is replaced by the maximum element to its right, and the last element remains unchanged.

Refer to the sample output for formatting specifications.

#### **Sample Test Case**

Input: 6  
12 3 91 15 12 14  
Output: 91 91 15 14 14 14

### Answer

```
// You are using Java
import java.util.Scanner;

public class Main {
    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();
        }

        // We'll traverse from right to left
        int maxFromRight = arr[n - 1]; // last element stays unchanged

        for (int i = n - 2; i >= 0; i--) {
            int current = arr[i];
            arr[i] = maxFromRight;
            if (current > maxFromRight) {
                maxFromRight = current;
            }
        }

        // Print the modified array
        for (int i = 0; i < n; i++) {
            System.out.print(arr[i]);
            if (i != n - 1) System.out.print(" ");
        }
        System.out.println();

        sc.close();
    }
}
```

Status : Correct

Marks : 10/10

# Rajalakshmi Engineering College

Name: Gnaana Kumaar

Email: 241001062@rajalakshmi.edu.in

Roll no: 241001062

Phone: 9790249960

Branch: REC

Department: IT - Section 4

Batch: 2028

Degree: B.E - IT

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**

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***

```
// You are using Java
import java.util.Scanner;

public class Main {

    // Helper method to check if a number is prime
    public static boolean isPrime(int num) {
        if (num <= 1) return false;
        if (num == 2) return true;
        if (num % 2 == 0) return false;

        for (int i = 3; i <= Math.sqrt(num); i += 2) {
            if (num % i == 0) return false;
        }
        return true;
    }

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

        // Input array size
        int n = sc.nextInt();
        int[] arr = new int[n];
```

```

// Input array elements
int totalSum = 0;
for (int i = 0; i < n; i++) {
    arr[i] = sc.nextInt();
    totalSum += arr[i];
}

boolean found = false;

// Try removing each element and check if remaining sum is prime
for (int i = 0; i < n; i++) {
    int sumAfterRemoval = totalSum - arr[i];
    if (isPrime(sumAfterRemoval)) {
        System.out.println(sumAfterRemoval);
        found = true;
        break;
    }
}

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

```

**Status : Correct**

**Marks : 10/10**

## 2. 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***

```
// You are using Java
import java.util.Scanner;

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

        // Read number of rows and columns
        int n = sc.nextInt(); // number of rows
        int m = sc.nextInt(); // number of columns

        int sum = 0;

        // Read the grid and compute sum
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                sum += sc.nextInt();
            }
        }
    }
}
```

```
        }  
    }  
  
    // Print the final sum  
    System.out.println(sum);  
}  
}
```

**Status : Correct**

**Marks : 10/10**

### 3. 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  $R1 \times C1$  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  $R2 \times C2$  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**

```
// You are using Java
import java.util.Scanner;

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

        // Read dimensions of first matrix
        int r1 = sc.nextInt();
        int c1 = sc.nextInt();
        int[][] mat1 = new int[r1][c1];

        // Read elements of first matrix
        for (int i = 0; i < r1; i++) {
            for (int j = 0; j < c1; j++) {
                mat1[i][j] = sc.nextInt();
            }
        }

        // Read dimensions of second matrix
        int r2 = sc.nextInt();
        int c2 = sc.nextInt();
        int[][] mat2 = new int[r2][c2];

        // Read elements of second matrix
        for (int i = 0; i < r2; i++) {
            for (int j = 0; j < c2; j++) {
                mat2[i][j] = sc.nextInt();
            }
        }
```

```

    }

    // Check if multiplication is possible
    if (c1 != r2) {
        System.out.println("Matrix multiplication not possible");
        return;
    }

    // Initialize result matrix
    int[][] result = new int[r1][c2];

    // Perform matrix multiplication
    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c2; j++) {
            for (int k = 0; k < c1; k++) {
                result[i][j] += mat1[i][k] * mat2[k][j];
            }
        }
    }

    // Print the result matrix
    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c2; j++) {
            System.out.print(result[i][j] + " ");
        }
        System.out.println();
    }
}

```

**Status :** Correct

**Marks :** 10/10

#### 4. 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***

```
// You are using Java
import java.util.Scanner;

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

        // Read number of reward points
        int n = sc.nextInt();
```

```
int[] points = new int[n];
// Read reward points (sorted)
for (int i = 0; i < n; i++) {
    points[i] = sc.nextInt();
}

// Process to remove duplicates
// Since array is sorted, duplicates will be adjacent
StringBuilder uniquePoints = new StringBuilder();
int count = 0;

for (int i = 0; i < n; i++) {
    // Print first element or only if different from previous
    if (i == 0 || points[i] != points[i - 1]) {
        uniquePoints.append(points[i]).append(" ");
        count++;
    }
}

// Print the unique points
System.out.print(uniquePoints.toString());

// Print count on the same line separated by two spaces as per sample
// output
System.out.println(" " + count);
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Gnaana Kumaar

Email: 241001062@rajalakshmi.edu.in

Roll no: 241001062

Phone: 9790249960

Branch: REC

Department: IT - Section 4

Batch: 2028

Degree: B.E - IT

Scan to verify results



## 2024\_28\_III\_OOPS Using Java Lab

### 2028\_REC\_OOPS using Java\_Week 3\_Q5

Attempt : 1

Total Mark : 10

Marks Obtained : 10

#### **Section 1 : Coding**

##### **1. Problem Statement**

Sharon is creating a program that finds the first repeated element in an integer array. The program should efficiently identify the first element that appears more than once in the given array. If no such element is found, it should appropriately display a message.

Help Sharon to complete the program.

##### ***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 a repeated element is found, print the first element that appears more than once.

If no repeated element is found, print "No repeated element found in the array".

Refer to the sample output for formatting specifications.

### ***Sample Test Case***

Input: 8  
12 21 13 14 21 36 47 21

Output: 21

### ***Answer***

```
// You are using Java
import java.util.HashSet;
import java.util.Scanner;

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

        // Read number of elements
        int n = sc.nextInt();

        // Read array elements
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        // Use HashSet to track seen elements
        HashSet<Integer> seen = new HashSet<>();
        boolean found = false;

        for (int i = 0; i < n; i++) {
            if (seen.contains(arr[i])) {
                System.out.println(arr[i]);
            } else {
                seen.add(arr[i]);
            }
        }
    }
}
```

```
        found = true;
        break;
    } else {
        seen.add(arr[i]);
    }
}

// If no repeated element is found
if (!found) {
    System.out.println("No repeated element found in the array");
}
}
```

**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Gnaana Kumaar

Email: 241001062@rajalakshmi.edu.in

Roll no: 241001062

Phone: 9790249960

Branch: REC

Department: IT - Section 4

Batch: 2028

Degree: B.E - IT

Scan to verify results



## 2024\_28\_III\_OOPS Using Java Lab

### 2028\_REC\_OOPS using Java\_Week 3\_Q4

Attempt : 1

Total Mark : 10

Marks Obtained : 0

#### **Section 1 : Coding**

##### **1. Problem Statement**

Sesha is developing a weather monitoring system for a region with multiple weather stations. Each weather station collects temperature data hourly and stores it in a 2D array.

Write a program that can add the temperature data from two different weather stations to create a combined temperature record for the region.

##### ***Input Format***

The first line of input consists of two space-separated integers N and M, representing the number of rows and columns of the matrices, respectively.

The next N lines consist of M space-separated integers, representing the values of the first matrix.

The following N lines consist of M space-separated integers, representing the values of the second matrix.

#### ***Output Format***

The output prints the addition of the two matrices in N rows and M columns, representing the combined temperature record.

Refer to the sample output for formatting specifications.

#### ***Sample Test Case***

Input: 3 3

1 2 3

4 5 6

7 8 9

1 1 1

2 2 2

3 3 3

Output: 2 3 4

6 7 8

10 11 12

#### ***Answer***

```
// You are using Java
import java.util.Scanner;

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

        // Read number of rows and columns
        int n = sc.nextInt();
        int m = sc.nextInt();

        // Create matrices
        int[][] matrix1 = new int[n][m];
        int[][] matrix2 = new int[n][m];
        int[][] result = new int[n][m];

        // Read first matrix
```

```
for (int i = 0; i < n; i++) {  
    for (int j = 0; j < m; j++) {  
        matrix1[i][j] = sc.nextInt();  
    }  
}  
  
// Read second matrix  
for (int i = 0; i < n; i++) {  
    for (int j = 0; j < m; j++) {  
        matrix2[i][j] = sc.nextInt();  
    }  
}  
  
// Add both matrices  
for (int i = 0; i < n; i++) {  
    for (int j = 0; j < m; j++) {  
        result[i][j] = matrix1[i][j] + matrix2[i][j];  
    }  
}  
  
// Print the result matrix  
for (int i = 0; i < n; i++) {  
    for (int j = 0; j < m; j++) {  
        System.out.print(result[i][j] + " ");  
    }  
    System.out.println(); // Move to next row  
}
```

Status : Wrong

Marks : 0/10

# Rajalakshmi Engineering College

Name: Gnaana Kumaar

Email: 241001062@rajalakshmi.edu.in

Roll no: 241001062

Phone: 9790249960

Branch: REC

Department: IT - Section 4

Batch: 2028

Degree: B.E - IT

Scan to verify results



## 2024\_28\_III\_OOPS Using Java Lab

### 2028\_REC\_OOPS using Java\_Week 3\_Q3

Attempt : 1

Total Mark : 10

Marks Obtained : 10

#### **Section 1 : Coding**

##### **1. Problem Statement**

You are developing a warehouse management system for a shipping company. The system uses an integer array to represent the weights of packages in a specific order. To verify that the weight capacity is not exceeded, the program needs to calculate the sum of the weights of the first and last packages in the list.

Task:

Write a code to calculate the sum of the weights of the first and last packages in the list. The program should take an integer array as input and return the total weight of the first and last packages.

##### ***Input Format***

The first line of the input is an integer N representing the size of the array.

The second line of the input is N space-separated integer values.

#### ***Output Format***

The output is displayed in the following format:

"Sum of the first and last elements: <>Sum<>"

Refer to the sample output for formatting specifications.

#### ***Sample Test Case***

Input: 5

10 20 30 40 50

Output: Sum of the first and last elements: 60

#### ***Answer***

```
// You are using Java  
import java.util.Scanner;
```

```
class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        // Read size of the array  
        int n = sc.nextInt();  
  
        // Create array to store weights  
        int[] weights = new int[n];  
  
        // Read array elements  
        for (int i = 0; i < n; i++) {  
            weights[i] = sc.nextInt();  
        }  
  
        // Calculate sum of first and last elements  
        int sum = weights[0] + weights[n - 1];  
  
        // Output the result  
        System.out.println("Sum of the first and last elements: " + sum);
```

}

**Status : Correct**

241001062

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Gnaana Kumaar

Email: 241001062@rajalakshmi.edu.in

Roll no: 241001062

Phone: 9790249960

Branch: REC

Department: IT - Section 4

Batch: 2028

Degree: B.E - IT

Scan to verify results



## 2024\_28\_III\_OOPS Using Java Lab

### 2028\_REC\_OOPS using Java\_Week 3\_Q2

Attempt : 1

Total Mark : 10

Marks Obtained : 10

#### **Section 1 : Coding**

##### **1. Problem Statement**

Monica is interested in finding a treasure but the key to opening is to get the sum of the main diagonal elements and secondary diagonal elements.

Write a program to help Monica find the diagonal sum of a square 2D array.

Note: The main diagonal of the array consists of the elements traversing from the top-left corner to the bottom-right corner. The secondary diagonal includes elements from the top-right corner to the bottom-left corner.

##### ***Input Format***

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

The following N lines consist of N space-separated integers, representing the 2D array elements.

#### **Output Format**

The first line of output prints "Sum of the main diagonal: " followed by an integer, representing the sum of the main diagonal.

The second line prints "Sum of the secondary diagonal: " followed by an integer, representing the sum of the secondary diagonal.

Refer to the sample output for formatting specifications.

#### **Sample Test Case**

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

Output: Sum of the main diagonal: 15  
Sum of the secondary diagonal: 15

#### **Answer**

```
// You are using Java
import java.util.Scanner;
class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Read the size of the matrix
        int n = sc.nextInt();

        // Create the 2D array
        int[][] matrix = new int[n][n];

        // Read the matrix elements
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                matrix[i][j] = sc.nextInt();
            }
        }
    }
}
```

```
int mainDiagonalSum = 0;
int secondaryDiagonalSum = 0;

// Calculate the sums
for (int i = 0; i < n; i++) {
    mainDiagonalSum += matrix[i][i];           // Main diagonal: (0,0), (1,1), ...
    secondaryDiagonalSum += matrix[i][n - 1 - i]; // Secondary diagonal:
(0,n-1), (1,n-2), ...
}

// Output the results
System.out.println("Sum of the main diagonal: " + mainDiagonalSum);
System.out.println("Sum of the secondary diagonal: " +
secondaryDiagonalSum);
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Gnaana Kumaar

Email: 241001062@rajalakshmi.edu.in

Roll no: 241001062

Phone: 9790249960

Branch: REC

Department: IT - Section 4

Batch: 2028

Degree: B.E - IT

Scan to verify results



## 2024\_28\_III\_OOPS Using Java Lab

### 2028\_REC\_OOPS using Java\_Week 3\_Q1

Attempt : 1

Total Mark : 10

Marks Obtained : 10

#### **Section 1 : Coding**

##### **1. Problem Statement**

Rosh is intrigued by numerical patterns. Today, she stumbled upon a puzzle while working with arrays. She wants to compute the sum of the third-largest and second-smallest elements from a list of integers. She seeks your help to implement a program that solves this for her efficiently.

##### ***Input Format***

The first line of input is an integer N, representing the size of the array.

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

##### ***Output Format***

The output displays a single integer representing the sum of the third-largest and second-smallest elements in the array.

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: 10  
10 20 30 40 50 60 70 80 90 100  
Output: 100

### ***Answer***

```
// You are using Java
import java.util.*;

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

        // Read input size
        int n = scanner.nextInt();
        int[] arr = new int[n];

        // Read array elements
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }

        // Convert array to list and remove duplicates (if needed)
        List<Integer> list = new ArrayList<>();
        for (int num : arr) {
            list.add(num);
        }

        // Sort the list
        Collections.sort(list);

        // Get second-smallest and third-largest
        int secondSmallest = list.get(1);
        int thirdLargest = list.get(list.size() - 3);

        // Output the sum
```

```
        System.out.println(secondSmallest + thirdLargest);
    scanner.close();
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Gnaana Kumaar

Email: 241001062@rajalakshmi.edu.in

Roll no: 241001062

Phone: 9790249960

Branch: REC

Department: IT - Section 4

Batch: 2028

Degree: B.E - IT

Scan to verify results



## 2024\_28\_III\_OOPS Using Java Lab

### 2028\_REC\_OOPS using Java\_Week 2\_CY

Attempt : 1

Total Mark : 40

Marks Obtained : 40

#### **Section 1 : Coding**

##### **1. Problem Statement**

Joe has a favourite number, let's call it X. He wants to check if X is divisible by the sum of its digits. If it is, he considers it a lucky number. If not, he wants to find the closest smaller number, that is divisible by the sum of digits of X. Joe has challenged his friends to solve this puzzle at his birthday party.

##### **Example**

Input:

157

Output:

157 is not divisible by the sum of its digits.

The closest smaller number that is divisible: 156

**Explanation:**

The sum of the digits of X is  $1+5+7=13$ . Since 157 is not divisible by 13, we need to find the closest smaller number that is divisible by 13. 156 is divisible by 13, it is the closest smaller number that meets the requirement.

### ***Input Format***

The input consists of an integer X, representing Joe's favourite number.

### ***Output Format***

If X is a lucky number, then the output must be in the format: "X is divisible by the sum of its digits."

If not, then the output must be in the format:

"X is not divisible by the sum of its digits.

The closest smaller number that is divisible: Y",

where X is the entered number and Y is the closest number.

Refer to the sample output for formatting specifications.

### ***Sample Test Case***

Input: 120

Output: 120 is divisible by the sum of its digits.

### ***Answer***

```
// You are using Java
import java.util.Scanner;

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

        int sumOfDigits = 0;
```

```

int temp = X;

while (temp > 0) {
    sumOfDigits += temp % 10;
    temp /= 10;
}

if (X % sumOfDigits == 0) {
    System.out.print(X + " is divisible by the sum of its digits.");
} else {
    int closest = X - 1;
    while (closest > 0) {
        if (closest % sumOfDigits == 0) {
            break;
        }
        closest--;
    }
    System.out.print(X + " is not divisible by the sum of its digits. The closest
smaller number that is divisible: " + closest);
}
}

```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Raj is solving a physics problem involving projectile motion, where he needs to calculate the time a ball hits the ground using a quadratic equation of the form  $ax^2 + bx + c = 0$ . Depending on the coefficients, the ball may hit the ground once, twice, or not at all in real time.

Help Raj find all real roots of the equation, if any.

Note: discriminant =  $b^2 - 4ac$

### ***Input Format***

The input consists of three space-separated doubles  $a$ ,  $b$ , and  $c$ , representing the coefficients of the quadratic equation.

### **Output Format**

If there are two real roots, print:

- "Two real solutions:"
- "Root1 = <value>"
- "Root2 = <value>"

If there is one real root, print:

- "One real solution:"
- "Root = <value>"

If there are no real roots, print:

- "There are no real solutions."

Note: values are rounded to two decimal places.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 1 6 9

Output: One real solution:

Root = -3.00

### **Answer**

```
// You are using Java
import java.util.Scanner;

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

        double a = sc.nextDouble();
        double b = sc.nextDouble();
        double c = sc.nextDouble();

        double discriminant = b * b - 4 * a * c;
```

```
if (discriminant > 0) {  
    double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);  
    double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);  
    System.out.printf("Two real solutions: Root1 = %.2f Root2 = %.2f", root1,  
root2);  
} else if (discriminant == 0) {  
    double root = -b / (2 * a);  
    System.out.printf("One real solution: Root = %.2f", root);  
} else {  
    System.out.print("There are no real solutions.");  
}  
}  
}  
}
```

Status : Correct

Marks : 10/10

### 3. Problem Statement

John is a fitness trainer, and he wants to use the BMI calculator to assess the body mass index of his clients. He has a list of clients based on their height and weight.

John plans to write a program to quickly determine the BMI and provide a classification for each client.

If BMI is less than 18.5, the program will classify it as "Underweight" If BMI is between 18.6 and 24.9, the program will classify it as "Normal Weight" If BMI is between 25.0 and 29.9, the program will classify it as "Overweight" If BMI is 30.0 or higher, the program will classify it as "Obese"

Note: Formula to calculate BMI = weight/(height\*height)

#### ***Input Format***

The first line of input consists of a double value, representing the height of the person in meters.

The second line consists of a double value, representing the weight of the person in kilograms.

#### ***Output Format***

The first line of output prints "BMI: " followed by a double (rounded to two decimal places) representing the calculated BMI.

The second line prints "Classification: " followed by a string indicating the BMI category (Underweight, Normal Weight, Overweight, or Obese).

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 1.2

45.2

Output: BMI: 31.39

Classification: Obese

### **Answer**

```
// You are using Java  
import java.util.Scanner;
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        double height = sc.nextDouble();  
        double weight = sc.nextDouble();  
  
        double bmi = weight / (height * height);  
  
        System.out.printf("BMI: %.2f ", bmi);  
  
        if (bmi < 18.5) {  
            System.out.print("Classification: Underweight");  
        } else if (bmi >= 18.6 && bmi <= 24.9) {  
            System.out.print("Classification: Normal Weight");  
        } else if (bmi >= 25.0 && bmi <= 29.9) {  
            System.out.print("Classification: Overweight");  
        } else if (bmi >= 30.0) {  
            System.out.print("Classification: Obese");  
        }  
    }  
}
```

}

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Ted, the computer science enthusiast, has accepted the challenge of writing a program that checks if the number of digits in an integer matches the sum of its digits.

Guide Ted in designing and writing the code to solve this problem using a 'do-while' loop.

##### ***Input Format***

The input consists of an integer N, representing the number to be checked.

##### ***Output Format***

If the sum is equal to the number of digits, print "The number of digits in N matches the sum of its digits."

Else, print "The number of digits in N does not match the sum of its digits."

Refer to the sample output for formatting specifications.

##### ***Sample Test Case***

Input: 20

Output: The number of digits in 20 matches the sum of its digits.

##### ***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 original = N;
```

```
int digitCount = 0;
int digitSum = 0;

do {
    int digit = N % 10;
    digitSum += digit;
    digitCount++;
    N /= 10;
} while (N > 0);

if (digitCount == digitSum) {
    System.out.print("The number of digits in " + original + " matches the sum
of its digits.");
} else {
    System.out.print("The number of digits in " + original + " does not match
the sum of its digits.");
}
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Gnaana Kumaar

Email: 241001062@rajalakshmi.edu.in

Roll no: 241001062

Phone: 9790249960

Branch: REC

Department: IT - Section 4

Batch: 2028

Degree: B.E - IT

Scan to verify results



## 2024\_28\_III\_OOPS Using Java Lab

## 2028\_REC\_OOPS using Java\_Week 2\_CY

Attempt : 1

Total Mark : 40

Marks Obtained : 40

### **Section 1 : Coding**

#### **1. Problem Statement**

Joe has a favourite number, let's call it X. He wants to check if X is divisible by the sum of its digits. If it is, he considers it a lucky number. If not, he wants to find the closest smaller number, that is divisible by the sum of digits of X. Joe has challenged his friends to solve this puzzle at his birthday party.

#### **Example**

Input:

157

Output:

157 is not divisible by the sum of its digits.

The closest smaller number that is divisible: 156

**Explanation:**

The sum of the digits of X is  $1+5+7=13$ . Since 157 is not divisible by 13, we need to find the closest smaller number that is divisible by 13. 156 is divisible by 13, it is the closest smaller number that meets the requirement.

### ***Input Format***

The input consists of an integer X, representing Joe's favourite number.

### ***Output Format***

If X is a lucky number, then the output must be in the format: "X is divisible by the sum of its digits."

If not, then the output must be in the format:

"X is not divisible by the sum of its digits.

The closest smaller number that is divisible: Y",

where X is the entered number and Y is the closest number.

Refer to the sample output for formatting specifications.

### ***Sample Test Case***

Input: 120

Output: 120 is divisible by the sum of its digits.

### ***Answer***

```
// You are using Java
import java.util.Scanner;

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

        int sumOfDigits = 0;
```

```

int temp = X;

while (temp > 0) {
    sumOfDigits += temp % 10;
    temp /= 10;
}

if (X % sumOfDigits == 0) {
    System.out.print(X + " is divisible by the sum of its digits.");
} else {
    int closest = X - 1;
    while (closest > 0) {
        if (closest % sumOfDigits == 0) {
            break;
        }
        closest--;
    }
    System.out.print(X + " is not divisible by the sum of its digits. The closest
smaller number that is divisible: " + closest);
}
}

```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Raj is solving a physics problem involving projectile motion, where he needs to calculate the time a ball hits the ground using a quadratic equation of the form  $ax^2 + bx + c = 0$ . Depending on the coefficients, the ball may hit the ground once, twice, or not at all in real time.

Help Raj find all real roots of the equation, if any.

Note: discriminant =  $b^2 - 4ac$

### ***Input Format***

The input consists of three space-separated doubles  $a$ ,  $b$ , and  $c$ , representing the coefficients of the quadratic equation.

### **Output Format**

If there are two real roots, print:

- "Two real solutions:"
- "Root1 = <value>"
- "Root2 = <value>"

If there is one real root, print:

- "One real solution:"
- "Root = <value>"

If there are no real roots, print:

- "There are no real solutions."

Note: values are rounded to two decimal places.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 1 6 9

Output: One real solution:

Root = -3.00

### **Answer**

```
// You are using Java
import java.util.Scanner;

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

        double a = sc.nextDouble();
        double b = sc.nextDouble();
        double c = sc.nextDouble();

        double discriminant = b * b - 4 * a * c;
```

```
if (discriminant > 0) {  
    double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);  
    double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);  
    System.out.printf("Two real solutions: Root1 = %.2f Root2 = %.2f", root1,  
root2);  
} else if (discriminant == 0) {  
    double root = -b / (2 * a);  
    System.out.printf("One real solution: Root = %.2f", root);  
} else {  
    System.out.print("There are no real solutions.");  
}  
}  
}  
}
```

Status : Correct

Marks : 10/10

### 3. Problem Statement

John is a fitness trainer, and he wants to use the BMI calculator to assess the body mass index of his clients. He has a list of clients based on their height and weight.

John plans to write a program to quickly determine the BMI and provide a classification for each client.

If BMI is less than 18.5, the program will classify it as "Underweight" If BMI is between 18.6 and 24.9, the program will classify it as "Normal Weight" If BMI is between 25.0 and 29.9, the program will classify it as "Overweight" If BMI is 30.0 or higher, the program will classify it as "Obese"

Note: Formula to calculate BMI = weight/(height\*height)

#### ***Input Format***

The first line of input consists of a double value, representing the height of the person in meters.

The second line consists of a double value, representing the weight of the person in kilograms.

#### ***Output Format***

The first line of output prints "BMI: " followed by a double (rounded to two decimal places) representing the calculated BMI.

The second line prints "Classification: " followed by a string indicating the BMI category (Underweight, Normal Weight, Overweight, or Obese).

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 1.2

45.2

Output: BMI: 31.39

Classification: Obese

### **Answer**

```
// You are using Java
import java.util.Scanner;

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

        double height = sc.nextDouble();
        double weight = sc.nextDouble();

        double bmi = weight / (height * height);

        System.out.printf("BMI: %.2f ", bmi);

        if (bmi < 18.5) {
            System.out.print("Classification: Underweight");
        } else if (bmi >= 18.6 && bmi <= 24.9) {
            System.out.print("Classification: Normal Weight");
        } else if (bmi >= 25.0 && bmi <= 29.9) {
            System.out.print("Classification: Overweight");
        } else if (bmi >= 30.0) {
            System.out.print("Classification: Obese");
        }
    }
}
```

}

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Ted, the computer science enthusiast, has accepted the challenge of writing a program that checks if the number of digits in an integer matches the sum of its digits.

Guide Ted in designing and writing the code to solve this problem using a 'do-while' loop.

##### ***Input Format***

The input consists of an integer N, representing the number to be checked.

##### ***Output Format***

If the sum is equal to the number of digits, print "The number of digits in N matches the sum of its digits."

Else, print "The number of digits in N does not match the sum of its digits."

Refer to the sample output for formatting specifications.

##### ***Sample Test Case***

Input: 20

Output: The number of digits in 20 matches the sum of its digits.

##### ***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 original = N;
```

```
int digitCount = 0;
int digitSum = 0;

do {
    int digit = N % 10;
    digitSum += digit;
    digitCount++;
    N /= 10;
} while (N > 0);

if (digitCount == digitSum) {
    System.out.print("The number of digits in " + original + " matches the sum
of its digits.");
} else {
    System.out.print("The number of digits in " + original + " does not match
the sum of its digits.");
}
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Gnaana Kumaar

Email: 241001062@rajalakshmi.edu.in

Roll no: 241001062

Phone: 9790249960

Branch: REC

Department: IT - Section 4

Batch: 2028

Degree: B.E - IT

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**

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***

```
// You are using Java
import java.util.Scanner;

public class Main {

    // Helper method to check if a number is prime
    public static boolean isPrime(int num) {
        if (num <= 1) return false;
        if (num == 2) return true;
        if (num % 2 == 0) return false;

        for (int i = 3; i <= Math.sqrt(num); i += 2) {
            if (num % i == 0) return false;
        }
        return true;
    }

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

        // Input array size
        int n = sc.nextInt();
        int[] arr = new int[n];
```

```
// Input array elements
int totalSum = 0;
for (int i = 0; i < n; i++) {
    arr[i] = sc.nextInt();
    totalSum += arr[i];
}

boolean found = false;

// Try removing each element and check if remaining sum is prime
for (int i = 0; i < n; i++) {
    int sumAfterRemoval = totalSum - arr[i];
    if (isPrime(sumAfterRemoval)) {
        System.out.println(sumAfterRemoval);
        found = true;
        break;
    }
}

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

Status : Correct

Marks : 10/10

## 2. 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***

```
// You are using Java
import java.util.Scanner;

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

        // Read number of rows and columns
        int n = sc.nextInt(); // number of rows
        int m = sc.nextInt(); // number of columns

        int sum = 0;

        // Read the grid and compute sum
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                sum += sc.nextInt();
            }
        }
    }
}
```

```
        }  
    }  
  
    // Print the final sum  
    System.out.println(sum);  
}  
}
```

**Status : Correct**

**Marks : 10/10**

### 3. 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  $R1 \times C1$  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  $R2 \times C2$  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**

```
// You are using Java
import java.util.Scanner;

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

        // Read dimensions of first matrix
        int r1 = sc.nextInt();
        int c1 = sc.nextInt();
        int[][] mat1 = new int[r1][c1];

        // Read elements of first matrix
        for (int i = 0; i < r1; i++) {
            for (int j = 0; j < c1; j++) {
                mat1[i][j] = sc.nextInt();
            }
        }

        // Read dimensions of second matrix
        int r2 = sc.nextInt();
        int c2 = sc.nextInt();
        int[][] mat2 = new int[r2][c2];

        // Read elements of second matrix
        for (int i = 0; i < r2; i++) {
            for (int j = 0; j < c2; j++) {
                mat2[i][j] = sc.nextInt();
            }
        }
```

```

    }

    // Check if multiplication is possible
    if (c1 != r2) {
        System.out.println("Matrix multiplication not possible");
        return;
    }

    // Initialize result matrix
    int[][] result = new int[r1][c2];

    // Perform matrix multiplication
    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c2; j++) {
            for (int k = 0; k < c1; k++) {
                result[i][j] += mat1[i][k] * mat2[k][j];
            }
        }
    }

    // Print the result matrix
    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c2; j++) {
            System.out.print(result[i][j] + " ");
        }
        System.out.println();
    }
}

```

**Status :** Correct

**Marks :** 10/10

#### 4. 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***

```
// You are using Java
import java.util.Scanner;

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

        // Read number of reward points
        int n = sc.nextInt();
```

```
int[] points = new int[n];
// Read reward points (sorted)
for (int i = 0; i < n; i++) {
    points[i] = sc.nextInt();
}

// Process to remove duplicates
// Since array is sorted, duplicates will be adjacent
StringBuilder uniquePoints = new StringBuilder();
int count = 0;

for (int i = 0; i < n; i++) {
    // Print first element or only if different from previous
    if (i == 0 || points[i] != points[i - 1]) {
        uniquePoints.append(points[i]).append(" ");
        count++;
    }
}

// Print the unique points
System.out.print(uniquePoints.toString());

// Print count on the same line separated by two spaces as per sample
// output
System.out.println(" " + count);
}
```

**Status :** Correct

**Marks :** 10/10