Array coding question :

1. Find the Largest and Smallest Element

○ Given an array, find the smallest and largest elements in it.

```
class MinMaxNo{

            public static void main( String Args[]){

            int arr[]={54,23,56,5,80,30,15};

            int Max= arr[0];

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

                        if(Max < arr[i]){

                        Max=arr[i];

                        }

                        System.out.println("Maximum number of an array is = "+Max);

            int Min =arr[0];

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

                        if(Min> arr[i]){

                                    Min=arr[i];

                        }

                        System.out.println("Minimum number of an array is =" +Min);

            }

}
```

2. Reverse an Array

○ Reverse the given array in place.

```
public class ReverseArray {

   public static void reverseArray(int[] arr) {

      int left = 0, right = arr.length - 1;


      while (left < right) {

         // Swap elements

         int temp = arr[left];

         arr[left] = arr[right];

         arr[right] = temp;
```

```
            // Move pointers
            left++;
            right--;
        }
    }

    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};

        System.out.println("Original Array: " + Arrays.toString(arr));

        reverseArray(arr);

        System.out.println("Reversed Array: " + Arrays.toString(arr));
    }
}
```

3. Find the Second Largest Element

- Find the second-largest element in the given array.

```
public class SecondLargest {
    public static int findSecondLargest(int[] arr) {
        int n = arr.length;
        int largest = 0;
        for (int i = 1; i < n; i++) {
            if (arr[i] > arr[largest])
                largest = i;
        }

        int secondLargest = -1;
        for (int i = 0; i < n; i++) {
            if (arr[i] != arr[largest]) {
                if (secondLargest == -1 || arr[i] > arr[secondLargest])
                    secondLargest = i;
            }
        }

        if (secondLargest == -1)
            return -1; // No second largest element
        return arr[secondLargest];
    }

    public static void main(String[] args) {
        int[] arr = {10, 5, 8, 20, 15};
        int secondLargest = findSecondLargest(arr);
        if (secondLargest == -1)
            System.out.println("No second largest element");
        else
            System.out.println("Second largest element: " + secondLargest);
    }
}
```

4. Count Even and Odd Numbers

○ Count the number of even and odd numbers in an array.

```
// Class to find sum of even and odd numbers
class Numbers {
    public static void AddEven(int arr[]) {
        int SumEven = 0;  // Variable to store sum of even numbers
        int SumOdd = 0;   // Variable to store sum of odd numbers

        for (int i = 0; i < arr.length; i++) { // Corrected loop syntax
            if (arr[i] % 2 == 0) { // Check if even
                SumEven += arr[i];
            } else { // If odd
                SumOdd += arr[i];
            }
        }

        System.out.println("Sum of Even Numbers is: " + SumEven);
        System.out.println("Sum of Odd Numbers is: " + SumOdd);
    }
}

// Main class to test the function
public class EvenOddAddition {
    public static void main(String args[]) {
        int arr[] = {5, 8, 7, 10, 9, 52}; // Array input
        Numbers.AddEven(arr); // Corrected function call
    }
}
```

5. Find Sum and Average

○ Compute the sum and average of all elements in the array.

```
class Numbers {
   public static void AddAverage(int arr[]) {
      int Sum = 0;
      int Average = 0;

      for (int i = 0; i < arr.length; i++) {
         Sum += arr[i];
      Average = Sum / arr.length ;

      }

      System.out.println("Sum of all Numbers is: " + Sum);
      System.out.println("Average of all Numbers is : " + Average);
   }
}

// Main class to test the function
public class SumAverage {
   public static void main(String args[]) {
      int arr[] = {5, 8, 7, 10, 9, 52};
      Numbers.AddAverage(arr);
   }
}
```

6. Remove Duplicates from a Sorted Array

○ Remove duplicate elements from a sorted array without using extra space.

```
import java.util.Arrays;
class Duplicate {
   public static void Demo (int arr[]){
                        Arrays.sort(arr);
                        System.out.println("Updated Array without duplicates =");
                        int a =0;
                        for(int i = 1; i < arr.length ; i++){
                                    if(arr[i] != arr[i-1]){
                              a = arr[i];
                                                System.out.print(a + ", ");
                                    }
                                    a++;
                        }
            }
}

// Main class to test the function
public class DuplicateArray1 {
   public static void main(String args[]) {
       int arr[] = {5, 8, 7,6,7,5,5 ,10, 9, 52};
       Duplicate.Demo(arr);


   }
}
```

7. Rotate an Array

○ Rotate the array to the right by k positions.

```
import java.util.Arrays;
class Rotate{
        public static void done(int arr[], int k){
        int a = arr.length;
        k = k % a;
        int temp[] = new int[a];
                for(int i = 0; i < arr.length ; i++){
                                temp[(i = k) % a] = arr[i];
                        }
                for(int i = 0; i < arr.length; i++){
      arr[i] = temp[i];
                }
                System.out.println("Updated Array: " +Arrays.toString(arr));
        }
}
public class RotateDemo{
        public static void main(String args[]){
        int arr[] = {1,2,3,4,5,6,7,8};
        int k = 2;
        Rotate.done(arr,2);
        }
}
```

8. Merge Two Sorted Arrays

○ Merge two sorted arrays into a single sorted array without using extra space.

```
import java.util.Arrays;

class MergeDemo {
   public static void main(String args[]) {
      int arr1[] = {4, 5, 8, 9, 10, 20};
      int arr2[] = {1, 3, 7, 15, 12, 6};

      // Step 1: Create merged array of size arr1.length + arr2.length
      int mergedLength = arr1.length + arr2.length;
      int Sorted[] = new int[mergedLength];

      // Step 2: Copy arr1 and arr2 into Sorted array
      System.arraycopy(arr1, 0, Sorted, 0, arr1.length);
      System.arraycopy(arr2, 0, Sorted, arr1.length, arr2.length);

      // Step 3: Sort the merged array
      Arrays.sort(Sorted);

      // Step 4: Print the sorted merged array
      System.out.println("Sorted merged array: " + Arrays.toString(Sorted));
   }
}
```

9. Find Missing Number in an Array

○ Given an array of size n-1 containing numbers from 1 to n, find the missing number.

```
import java.util.Arrays;


class Missing{
	public static void Array(int arr[]){
		int n = arr.length;
		for(int i = 0; i < n ; i++){
			if(arr[i] != (i + 1)){
				System.out.println("Missing number is "+(i+1));
				break;
			}
		}
	}
}
public class MissingNumber{
	public static void main(String args[]){
		int arr[] = {1,2,3,5,6,7,8};
		Missing.Array(arr);
	}
}
```

10. Find Intersection and Union of Two Arrays

○ Find the intersection and union of two unsorted arrays.

```
import java.util.Arrays;


class UnionIntersection {


  // Function to find the Union of two sorted arrays
  public static void findUnion(int arr1[], int arr2[]) {
    int i = 0, j = 0;
    System.out.print("Union: ");

    while (i < arr1.length && j < arr2.length) {
        if (arr1[i] < arr2[j]) {
            System.out.print(arr1[i++] + " ");
        }
        else if (arr1[i] > arr2[j]) {
            System.out.print(arr2[j++] + " ");
        }
        else {  // Both elements are equal, take only one and move both pointers
            System.out.print(arr1[i] + " ");
            i++;
            j++;
        }
    }

    // Print remaining elements of arr1
    while (i < arr1.length) {
        System.out.print(arr1[i++] + " ");
    }
    // Print remaining elements of arr2
    while (j < arr2.length) {
        System.out.print(arr2[j++] + " ");
    }
    System.out.println();
}

// Function to find the Intersection of two sorted arrays
public static void findIntersection(int arr1[], int arr2[]) {
    int i = 0, j = 0;
    System.out.print("Intersection: ");
```

```
        while (i < arr1.length && j < arr2.length) {
            if (arr1[i] < arr2[j]) {
                i++;
            }
            else if (arr1[i] > arr2[j]) {
                j++;
            }
            else {  // Both are equal, add to intersection
                System.out.print(arr1[i] + " ");
                i++;
                j++;
            }
        }
        System.out.println();
    }

    public static void main(String args[]) {
        int arr1[] = {1, 2, 3, 5, 6, 7, 8};
        int arr2[] = {1, 2, 3, 4, 7, 8, 11};

        // Sort arrays first (if they are not already sorted)
        Arrays.sort(arr1);
        Arrays.sort(arr2);

        // Find Union and Intersection
        findUnion(arr1, arr2);
        findIntersection(arr1, arr2);
    }
}
```

11. Find a Subarray with Given Sum

○ Given an array of integers, find the subarray that sums to a given value S.

12. Write a program to accept 20 integer numbers in a single Dimensional Array. Find and Display the following:

○ Number of even numbers.

○ Number of odd numbers.

○ Number of multiples of 3

```
import java.util.Arrays;

import java.util.Scanner;


class EvenNumbers {

  public static void Even(int arr[]) {

    int count = 0; // Counter for even numbers

    System.out.print("Even Numbers: ");

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

      if (arr[i] % 2 == 0) {

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

        count++; // Increment count

      }

    }

    System.out.println("\nTotal Even Numbers: " + count); // Print count

  }

}


class OddNumbers {

  public static void Odd(int arr[]) {

    System.out.print("Odd Numbers: ");

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

      if (arr[i] % 2 != 0) {

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

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

class Divisible3 {
    public static void Divi(int arr[]) {
        System.out.print("Numbers Divisible by 3: ");
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] % 3 == 0) {
                System.out.print(arr[i] + " ");
            }
        }
        System.out.println();
    }
}

public class Calculator {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int arr[] = new int[5]; // Set to 5 for example, can be changed

        // Taking array input
        System.out.println("Enter " + arr.length + " numbers:");
        for (int i = 0; i < arr.length; i++) {
            arr[i] = sc.nextInt();
        }

        // Print array
        System.out.println("Provided Array Numbers: " + Arrays.toString(arr));
        // Call methods to find even, odd, and numbers divisible by 3
        EvenNumbers.Even(arr);
        OddNumbers.Odd(arr);
        Divisible3.Divi(arr);

        sc.close();
    }
}
```

13. Write a program to accept the marks in Physics, Chemistry and Maths secured by 20 class students in a single Dimensional Array. Find and display the following:

○ Number of students securing 75% and above in aggregate.

○ Number of students securing 40% and below in aggregate.

```
import java.util.Scanner;

public class StudentMarks {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int numStudents = 20;
        int totalMarks[] = new int[numStudents]; // Array to store total marks
        int countAbove75 = 0, countBelow40 = 0;

        // Taking input for 20 students
        for (int i = 0; i < numStudents; i++) {
            System.out.println("Enter marks for Student " + (i + 1) + " (Physics, Chemistry, Maths out of
100 each): ");
            int physics = sc.nextInt();
            int chemistry = sc.nextInt();
            int maths = sc.nextInt();

            // Calculate total marks (out of 300) and store in array
            totalMarks[i] = physics + chemistry + maths;

            // Calculate percentage
            double percentage = (totalMarks[i] / 300.0) * 100;

            // Counting students based on percentage
            if (percentage >= 75) {
                countAbove75++;
            } else if (percentage <= 40) {
                countBelow40++;
            }
        }

        // Display results
        System.out.println("\nNumber of students securing 75% and above: " + countAbove75);
        System.out.println("Number of students securing 40% and below: " + countBelow40);


    }
}
```

14. Write a program in Java to accept 20 numbers in a single dimensional array arr[20]. Transfer and store all the even numbers in an array even[ ] and all the odd numbers in another array odd[ ]. Finally, print the elements of the even & the odd array.

```
import java.util.Scanner;

public class EvenOddArrays {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int arr[] = new int[20]; // Array to store 20 numbers
        int even[] = new int[20]; // Array to store even numbers
        int odd[] = new int[20]; // Array to store odd numbers
        int evenCount = 0, oddCount = 0; // Counters for even and odd numbers

        // Taking input for 20 numbers
        System.out.println("Enter 20 numbers:");
        for (int i = 0; i < arr.length; i++) {
            arr[i] = sc.nextInt();

            // Separating even and odd numbers
            if (arr[i] % 2 == 0) {
                even[evenCount++] = arr[i]; // Store in even array
            } else {
                odd[oddCount++] = arr[i]; // Store in odd array
            }
        }

        // Print Even Numbers
        System.out.print("\nEven Numbers: ");
        for (int i = 0; i < evenCount; i++) {
            System.out.print(even[i] + " ");
        }
        // Print Odd Numbers
        System.out.print("\nOdd Numbers: ");
        for (int i = 0; i < oddCount; i++) {
            System.out.print(odd[i] + " ");
        }

        sc.close(); // Close scanner
    }
}
```

15. Write a Java program to print all sub-arrays with 0 sum present in a given array of integers.

Example:

Input :

nums1 = { 1, 3, -7, 3, 2, 3, 1, -3, -2, -2 }

nums2 = { 1, 2, -3, 4, 5, 6 }

nums3= { 1, 2, -2, 3, 4, 5, 6 }

Output:

```
Sub-arrays with 0 sum : [1, 3, -7, 3]
Sub-arrays with 0 sum : [3, -7, 3, 2, 3, 1, -3, -2]
Sub-arrays with 0 sum : [1, 2, -3]
Sub-arrays with 0 sum : [2, -2]
```

```
import java.util.ArrayList;
public class ZeroSumSubarrays {

    // Function to find and print all subarrays with sum 0
    public static void findZeroSumSubarrays(int[] arr) {
        int n = arr.length;
        boolean found = false;
    // Iterate through all possible subarrays
    for (int start = 0; start < n; start++) {
        int sum = 0;
        for (int end = start; end < n; end++) {
            sum += arr[end];

            // If sum is zero, print the subarray
            if (sum == 0) {
                found = true;
                printSubarray(arr, start, end);
            }
        }
    }

    if (!found) {
        System.out.println("No sub-array with sum 0 found.");
    }
}

// Function to print a subarray from start to end index
private static void printSubarray(int[] arr, int start, int end) {
    System.out.print("Sub-array with 0 sum: [");
    for (int i = start; i <= end; i++) {
        System.out.print(arr[i]);
        if (i < end) System.out.print(", ");
    }
    System.out.println("]");
}

// Main method
public static void main(String[] args) {
        int nums1[] = { 1, 3, -7, 3, 2, 3, 1, -3, -2, -2 };
        int nums2[] = { 1, 2, -3, 4, 5, 6 };
        int nums3[] = { 1, 2, -2, 3, 4, 5, 6 };

        System.out.println("For nums1:");
        findZeroSumSubarrays(nums1);

        System.out.println("\nFor nums2:");
        findZeroSumSubarrays(nums2);

        System.out.println("\nFor nums3:");
        findZeroSumSubarrays(nums3);
    }
}
```

16. Given two sorted arrays A and B of size p and q, write a Java program to merge elements of A with B by maintaining the sorted order i.e. fill A with first p smallest elements and fill B with remaining elements.

Example:

Input :

```
int[] A = { 1, 5, 6, 7, 8, 10 }
int[] B = { 2, 4, 9 }
```

Output:

```
Sorted Arrays:
A: [1, 2, 4, 5, 6, 7]
B: [8, 9, 10]
```

```java
import java.util.Arrays;

public class MergeSortedArrays {
  public static void mergeArrays(int[] A, int[] B) {
    int p = A.length;
    int q = B.length;
    int merged[] = new int[p + q];

    // Manually copying A and B into merged array
    for (int i = 0; i < p; i++) {
      merged[i] = A[i];
    }
    for (int i = 0; i < q; i++) {
      merged[p + i] = B[i];
    }

    // Sort the merged array
    Arrays.sort(merged);
```

```
        // Manually copying first p elements to A
        for (int i = 0; i < p; i++) {
            A[i] = merged[i];
        }

        // Manually copying remaining q elements to B
        for (int i = 0; i < q; i++) {
            B[i] = merged[p + i];
        }

        // Print the updated arrays
        System.out.println("Sorted Arrays:");
        System.out.println("A: " + Arrays.toString(A));
        System.out.println("B: " + Arrays.toString(B));
    }

    public static void main(String[] args) {
        int[] A = {1, 5, 6, 7, 8, 10};
        int[] B = {2, 4, 9};

        mergeArrays(A, B);
    }
}
```

17. Write a Java program to find the maximum product of two integers in a given array of integers.

Example:

Input :

nums = { 2, 3, 5, 7, -7, 5, 8, -5 }

Output:

Pair is (7, 8), Maximum Product: 56

```
import java.util.Arrays;

public class MergeSortedArrays {
    public static void mergeArrays(int[] A, int[] B) {
        int p = A.length;
        int q = B.length;
        int merged[] = new int[p + q];

        // Manually copying A and B into merged array
        for (int i = 0; i < p; i++) {
            merged[i] = A[i];
        }
        for (int i = 0; i < q; i++) {
            merged[p + i] = B[i];
        }

        // Sort the merged array
        Arrays.sort(merged);

        // Manually copying first p elements to A
        for (int i = 0; i < p; i++) {
            A[i] = merged[i];
        }

        // Manually copying remaining q elements to B
        for (int i = 0; i < q; i++) {
            B[i] = merged[p + i];
        }

        // Print the updated arrays
        System.out.println("Sorted Arrays:");
        System.out.println("A: " + Arrays.toString(A));
        System.out.println("B: " + Arrays.toString(B));
    }

    public static void main(String[] args) {
        int[] A = {1, 5, 6, 7, 8, 10};
        int[] B = {2, 4, 9};

        mergeArrays(A, B);
    }
}
```

18. Print a Matrix

○ Given an m x n matrix, print all its elements row-wise.

```
public class PrintMatrix {
  public static void printMatrix(int[][] matrix) {
    for (int i = 0; i < matrix.length; i++) { // Loop through rows
      for (int j = 0; j < matrix[i].length; j++) { // Loop through columns
        System.out.print(matrix[i][j] + " ");
      }
      System.out.println(); // Move to the next row
    }
  }

  public static void main(String[] args) {
    int[][] matrix = {
      {1, 2, 3},
      {4, 5, 6},
      {7, 8, 9}
    };

    System.out.println("Matrix printed row-wise:");
    printMatrix(matrix);
  }
}
```

19. Transpose of a Matrix

- Given a matrix, return its transpose (swap rows and columns).

```
public class TransposeMatrix {
    public static int[][] transpose(int[][] matrix) {
        int rows = matrix.length;
        int cols = matrix[0].length;
        int[][] transposed = new int[cols][rows]; // Swap rows and columns

        // Fill the transposed matrix
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                transposed[j][i] = matrix[i][j]; // Swap elements
            }
        }
        return transposed;
    }

    public static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int num : row) {
                System.out.print(num + " ");
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        int[][] matrix = {
            {1, 2, 3},
            {4, 5, 6}
        };
        System.out.println("Original Matrix:");
        printMatrix(matrix);

        int[][] transposedMatrix = transpose(matrix);

        System.out.println("\nTransposed Matrix:");
        printMatrix(transposedMatrix);
    }
}
```

20. Sum of Two Matrices

○ Given two matrices of the same size, compute their sum.

```
public class MatrixSum {
    public static int[][] addMatrices(int[][] A, int[][] B) {
        int rows = A.length;
        int cols = A[0].length;
        int[][] sum = new int[rows][cols];

        // Perform element-wise addition
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                sum[i][j] = A[i][j] + B[i][j];
            }
        }
        return sum;
    }

    public static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int num : row) {
                System.out.print(num + " ");
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        int[][] A = {
            {1, 2, 3},
            {4, 5, 6}
        };

        int[][] B = {
            {7, 8, 9},
            {10, 11, 12}
        };

        System.out.println("Matrix A:");
        printMatrix(A);

        System.out.println("\nMatrix B:");
        printMatrix(B);

        int[][] sumMatrix = addMatrices(A, B);

        System.out.println("\nSum of Matrices:");
        printMatrix(sumMatrix);
    }
}
```

21. Row-wise and Column-wise Sum

- Find the sum of each row and each column of a given matrix.

```
public class MatrixRowColumnSum {
    public static void rowColumnSum(int[][] matrix) {
        int rows = matrix.length;
        int cols = matrix[0].length;

        // Compute row sums
        System.out.println("Row-wise Sum:");
        for (int i = 0; i < rows; i++) {
            int rowSum = 0;
            for (int j = 0; j < cols; j++) {
                rowSum += matrix[i][j];
            }
            System.out.println("Row " + (i + 1) + " sum: " + rowSum);
        }

        // Compute column sums
        System.out.println("\nColumn-wise Sum:");
        for (int j = 0; j < cols; j++) {
            int colSum = 0;
            for (int i = 0; i < rows; i++) {
                colSum += matrix[i][j];
            }
            System.out.println("Column " + (j + 1) + " sum: " + colSum);
        }
    }

    public static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int num : row) {
                System.out.print(num + " ");
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        int[][] matrix = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        System.out.println("Matrix:");
        printMatrix(matrix);

        System.out.println();
        rowColumnSum(matrix);
    }
}
```

22. Find the Maximum Element in a Matrix

- Find the largest element in a given matrix.

```
public class MaxElementMatrix {
  public static int findMaxElement(int[][] matrix) {
    int maxElement = Integer.MIN_VALUE; // Initialize with the smallest possible integer

    // Iterate through the matrix to find the maximum element
    for (int i = 0; i < matrix.length; i++) {
      for (int j = 0; j < matrix[i].length; j++) {
        if (matrix[i][j] > maxElement) {
          maxElement = matrix[i][j];
        }
      }
    }

    return maxElement; // Return the maximum element found
  }

  public static void printMatrix(int[][] matrix) {
    for (int[] row : matrix) {
      for (int num : row) {
        System.out.print(num + " ");
      }
      System.out.println();
    }
  }

  public static void main(String[] args) {
    int[][] matrix = {
      {1, 4, 7},
      {8, 2, 6},
      {3, 9, 5}
    };

    System.out.println("Matrix:");
    printMatrix(matrix);

    int maxElement = findMaxElement(matrix);
    System.out.println("\nMaximum Element in the Matrix: " + maxElement);
  }
}
```

23. Matrix Multiplication

○ Multiply two matrices and return the resultant matrix.

```
public class MatrixMultiplication {
    public static int[][] multiplyMatrices(int[][] A, int[][] B) {
        int m = A.length;       // Rows of A
        int n = A[0].length;    // Columns of A (Rows of B)
        int p = B[0].length;    // Columns of B

        int[][] result = new int[m][p]; // Resultant matrix of size m x p

        // Multiplication logic
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < p; j++) {
                for (int k = 0; k < n; k++) {
                    result[i][j] += A[i][k] * B[k][j];
                }
            }
        }
        return result; // Returning the resultant matrix
    }

    public static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int num : row) {
                System.out.print(num + " ");
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        int[][] A = {
            {1, 2, 3},
            {4, 5, 6}
        };

        int[][] B = {
            {7, 8},
            {9, 10},
            {11, 12}
        };

        System.out.println("Matrix A:");
        printMatrix(A);

        System.out.println("\nMatrix B:");
        printMatrix(B);
        int[][] result = multiplyMatrices(A, B);

        System.out.println("\nResultant Matrix (A x B):");
        printMatrix(result);
    }
}
```

24. Rotate a Matrix by 90 Degrees

○ Rotate a given N x N matrix by 90 degrees clockwise.

```
public class RotateMatrix {
    public static void rotate90Clockwise(int[][] matrix) {
        int n = matrix.length;

        // Step 1: Transpose the matrix (Swap rows with columns)
        for (int i = 0; i < n; i++) {
            for (int j = i; j < n; j++) {
                int temp = matrix[i][j];
                matrix[i][j] = matrix[j][i];
                matrix[j][i] = temp;
            }
        }

        // Step 2: Reverse each row to get 90-degree rotation
        for (int i = 0; i < n; i++) {
            int left = 0, right = n - 1;
            while (left < right) {
                int temp = matrix[i][left];
                matrix[i][left] = matrix[i][right];
                matrix[i][right] = temp;
                left++;
                right--;
            }
        }
    }

    // Method to print the matrix
    public static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int num : row) {
                System.out.print(num + " ");
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        int[][] matrix = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        System.out.println("Original Matrix:");
        printMatrix(matrix);

        rotate90Clockwise(matrix);

        System.out.println("\nMatrix after 90-degree rotation:");
        printMatrix(matrix);
    }
}
```

25. Find the Diagonal Sum

○ Compute the sum of both diagonals in a square matrix.

```
public class DiagonalSum {
    public static int findDiagonalSum(int[][] matrix) {
        int n = matrix.length;
        int sum = 0;

        for (int i = 0; i < n; i++) {
            sum += matrix[i][i]; // Primary diagonal
            sum += matrix[i][n - 1 - i]; // Secondary diagonal
        }

        // If N is odd, subtract the middle element (counted twice)
        if (n % 2 == 1) {
            sum -= matrix[n / 2][n / 2];
        }

        return sum;
    }

    // Method to print the matrix
    public static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int num : row) {
                System.out.print(num + " ");
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        int[][] matrix = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        System.out.println("Given Matrix:");
        printMatrix(matrix);

        int diagonalSum = findDiagonalSum(matrix);
        System.out.println("\nSum of both diagonals: " + diagonalSum);
    }
}
```