**Array coding question :**

1. Find the Largest and Smallest Element

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

public class Main {

public static void main(String[] args) {

int[] arr = {23, 56, 12, 89, 2, 100, 43};

int[] result = findSmallestAndLargest(arr);

System.out.println("Smallest Element: " + result[0]);

System.out.println("Largest Element: " + result[1]);

}

public static int[] findSmallestAndLargest(int[] arr) {

if (arr == null || arr.length == 0) {

return new int[]{Integer.MAX\_VALUE, Integer.MIN\_VALUE}; // Handle empty array case

}

int smallest = arr[0];

int largest = arr[0];

for (int num : arr) {

if (num < smallest) {

smallest = num; // Update smallest

}

if (num > largest) {

largest = num; // Update largest

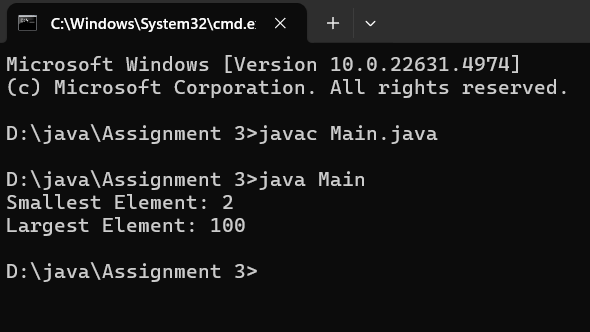
}

}

return new int[]{smallest, largest};

}

}



Reverse an Array

○ Reverse the given array in place.

public class Main {

public static void main(String[] args) {

int[] arr = {23, 56, 12, 89, 2, 100, 43};

System.out.println("Original Array:");

printArray(arr);

reverseArray(arr);

System.out.println("\nReversed Array:");

printArray(arr);

}

public static void reverseArray(int[] arr) {

int left = 0;

int right = arr.length - 1;

while (left < right) {

// Swap the elements at left and right

int temp = arr[left];

arr[left] = arr[right];

arr[right] = temp;

left++;

right--;

}

}

public static void printArray(int[] arr) {

for (int num : arr) {

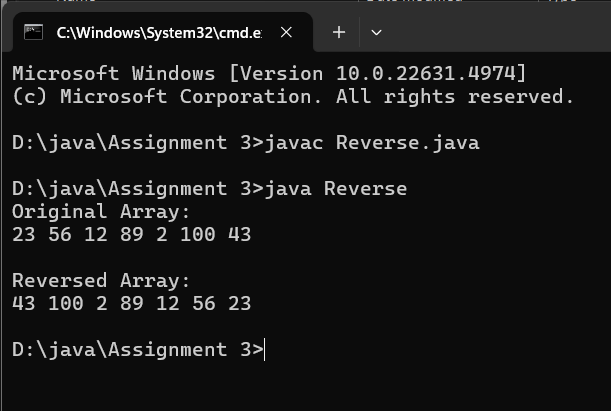
System.out.print(num + " ");

}

System.out.println();

}

}



Find the Second Largest Element

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

public class findSecondLargest {

public static void main(String[] args) {

int[] arr = {23, 56, 12, 89, 2, 100, 43};

int secondLargest = findSecondLargest(arr);

if (secondLargest != Integer.MIN\_VALUE) {

System.out.println("Second Largest Element: " + secondLargest);

} else {

System.out.println("Array doesn't have enough elements for second largest.");

}

}

public static int findSecondLargest(int[] arr) {

if (arr == null || arr.length < 2) {

return Integer.MIN\_VALUE; // If array has fewer than 2 elements, return a sentinel value

}

int largest = Integer.MIN\_VALUE;

int secondLargest = Integer.MIN\_VALUE;

for (int num : arr) {

if (num > largest) {

secondLargest = largest; // Update second largest before updating largest

largest = num; // Update largest

} else if (num > secondLargest && num != largest) {

secondLargest = num; // Update second largest

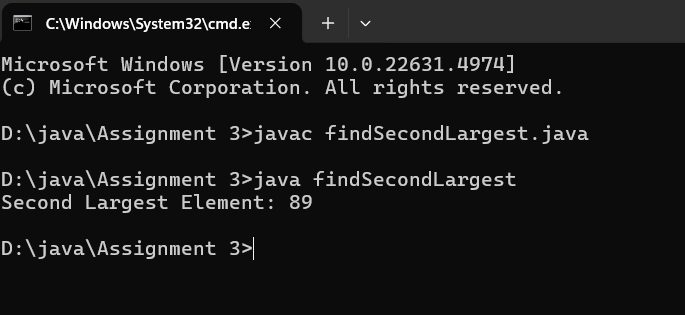
}

}

return secondLargest;

}

}



Count Even and Odd Numbers

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

public class EvenAndOdd {

public static void main(String[] args) {

int[] arr = {23, 56, 12, 89, 2, 100, 43, 65};

int[] result = countEvenAndOdd(arr);

System.out.println("Even Numbers Count: " + result[0]);

System.out.println("Odd Numbers Count: " + result[1]);

}

public static int[] countEvenAndOdd(int[] arr) {

int evenCount = 0;

int oddCount = 0;

for (int num : arr) {

if (num % 2 == 0) {

evenCount++; // Increment even count

} else {

oddCount++; // Increment odd count

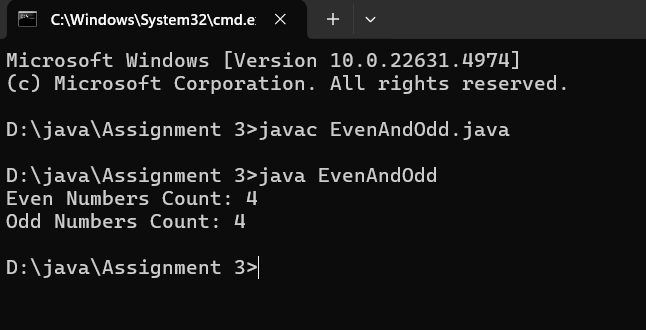
}

}

return new int[] {evenCount, oddCount};

}

}



Find Sum and Average

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

public class SumAndAverage {

public static void main(String[] args) {

int[] arr = {23, 56, 12, 89, 2, 100, 43};

double[] result = computeSumAndAverage(arr);

System.out.println("Sum of elements: " + result[0]);

System.out.println("Average of elements: " + result[1]);

}

public static double[] computeSumAndAverage(int[] arr) {

int sum = 0;

for (int num : arr) {

sum += num;

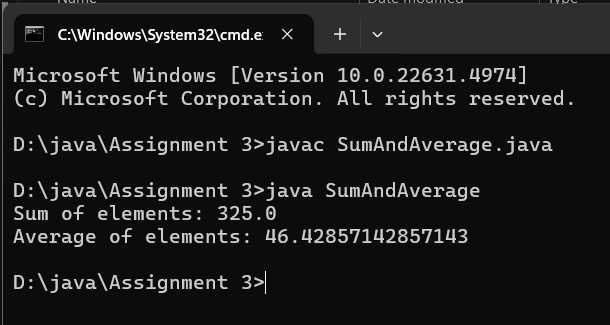
}

double average = (double) sum / arr.length;

return new double[] {sum, average};

}

}



Remove Duplicates from a Sorted Array

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

public class removeDuplicates {

public static void main(String[] args) {

int[] arr = {2, 2, 3, 3, 4, 5, 5, 6};

int newLength = removeDuplicates(arr);

System.out.println("Array after removing duplicates:");

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

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

}

}

public static int removeDuplicates(int[] arr) {

if (arr == null || arr.length == 0) {

return 0;

}

int uniqueIndex = 1

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

if (arr[i] != arr[i - 1]) {

arr[uniqueIndex] = arr[i];

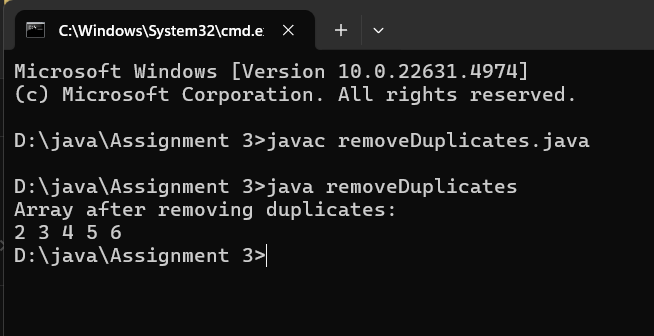
uniqueIndex++;

}

}

return uniqueIndex;

}

}

7. Rotate an Array

○ Rotate the array to the right by k positions.

public class rotateArray {

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4, 5, 6, 7};

int k = 3;

rotateArray(arr, k);

System.out.println("Array after rotation:");

for (int num : arr) {

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

}

}

public static void rotateArray(int[] arr, int k) {

int n = arr.length;

k = k % n;

if (k == 0) return;

reverse(arr, 0, n - 1);

reverse(arr, 0, k - 1);

reverse(arr, k, n - 1);

}

public static void reverse(int[] arr, int start, int end) {

while (start < end) {

int temp = arr[start];

arr[start] = arr[end];

arr[end] = temp;

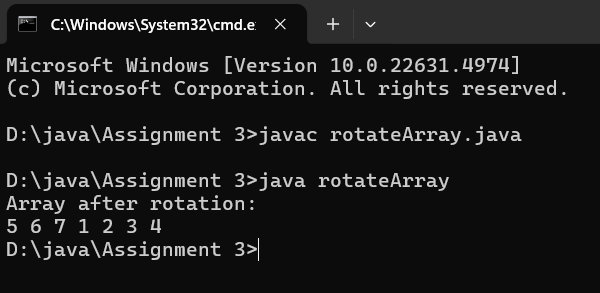
start++;

end--;

}

}

}



8. Merge Two Sorted Arrays

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

public class Main2 {

public static void main(String[] args) {

int[] arr1 = {1, 3, 5, 7};

int[] arr2 = {2, 4, 6, 8};

mergeSortedArrays(arr1, arr2);

System.out.println("Merged Array:");

for (int num : arr1) {

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

}

}

public static void mergeSortedArrays(int[] arr1, int[] arr2) {

int n = arr1.length;

int m = arr2.length;

int i = n - 1;

int j = m - 1;

int k = n + m - 1;

while (i >= 0 && j >= 0) {

if (arr1[i] > arr2[j]) {

arr1[k--] = arr1[i--];

} else {

arr1[k--] = arr2[j--];

}

}

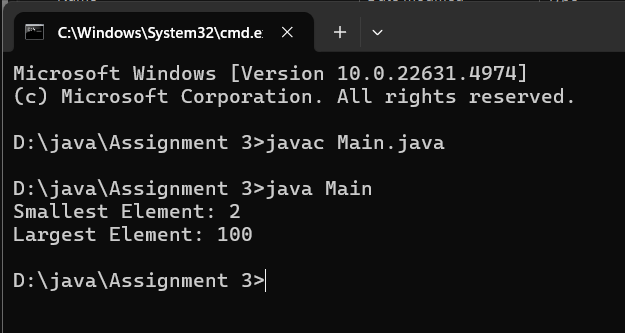
while (j >= 0) {

arr1[k--] = arr2[j--];

}

}

}



9. Find Missing Number in an Array

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

public class MissingNumber {

public static int findMissingNumber(int[] nums) {

int n = nums.length + 1; // Total numbers should be n

int expectedSum = n \* (n + 1) / 2; // Sum of first n natural numbers

int actualSum = 0;

for (int num : nums) {

actualSum += num;

}

return expectedSum - actualSum;

}

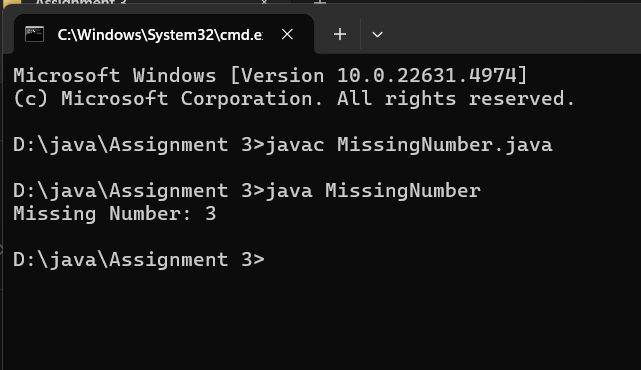
public static void main(String[] args) {

int[] nums = {1, 2, 4, 5, 6};

System.out.println("Missing Number: " + findMissingNumber(nums));

}

}



Find Intersection and Union of Two Arrays

○ Find the intersection and union of two unsorted arrays.

import java.util.HashSet;

import java.util.Set;

import java.util.Arrays;

public class ArrayIntersection {

public static int[] findIntersection(int[] array1, int[] array2) {

Set<Integer> set1 = new HashSet<>();

Set<Integer> intersection = new HashSet<>();

// Add elements of array1 to set1

for (int num : array1) {

set1.add(num);

}

// Check for common elements in array2

for (int num : array2) {

if (set1.contains(num)) {

intersection.add(num);

}

}

// Convert the intersection set to an array

int[] result = new int[intersection.size()];

int index = 0;

for (int num : intersection) {

result[index++] = num;

}

return result;

}

public static void main(String[] args) {

int[] array1 = {1, 2, 4, 5, 9};

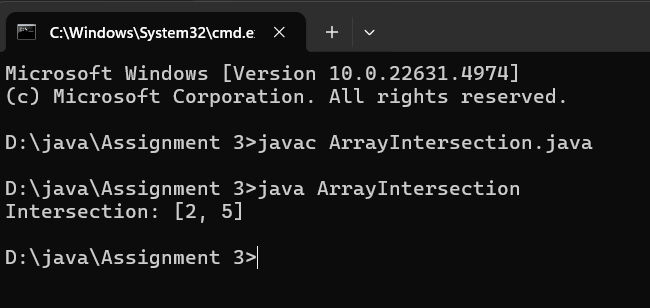
int[] array2 = {3, 2, 5, 7};

int[] intersection = findIntersection(array1, array2);

System.out.println("Intersection: " + Arrays.toString(intersection));

}

}



import java.util.HashSet;

import java.util.Set;

import java.util.Arrays;

public class ArrayUnion {

public static int[] findUnion(int[] array1, int[] array2) {

Set<Integer> unionSet = new HashSet<>();

// Add elements from both arrays to the unionSet

for (int num : array1) {

unionSet.add(num);

}

for (int num : array2) {

unionSet.add(num);

}

// Convert the union set to an array

int[] result = new int[unionSet.size()];

int index = 0;

for (int num : unionSet) {

result[index++] = num;

}

return result;

}

public static void main(String[] args) {

int[] array1 = {1, 2, 4, 5, 9};

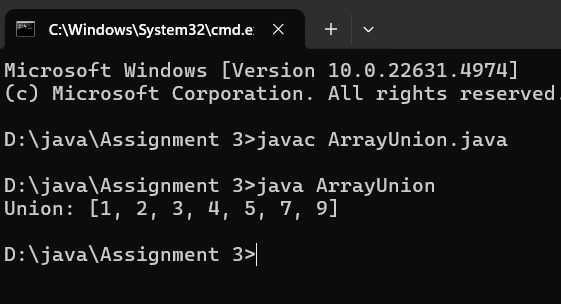
int[] array2 = {3, 2, 5, 7};

int[] union = findUnion(array1, array2);

System.out.println("Union: " + Arrays.toString(union));

}

}



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.Scanner;

public class ArrayAnalysis {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int[] numbers = new int[20];

int evenCount = 0, oddCount = 0, multipleOfThreeCount = 0;

// Input 20 integers

System.out.println("Please enter 20 integer numbers:");

for (int i = 0; i < 20; i++) {

numbers[i] = scanner.nextInt();

}

// Analyze the array

for (int num : numbers) {

if (num % 2 == 0) {

evenCount++;

} else {

oddCount++;

}

if (num % 3 == 0) {

multipleOfThreeCount++;

}

}

// Display the results

System.out.println("Number of even numbers: " + evenCount);

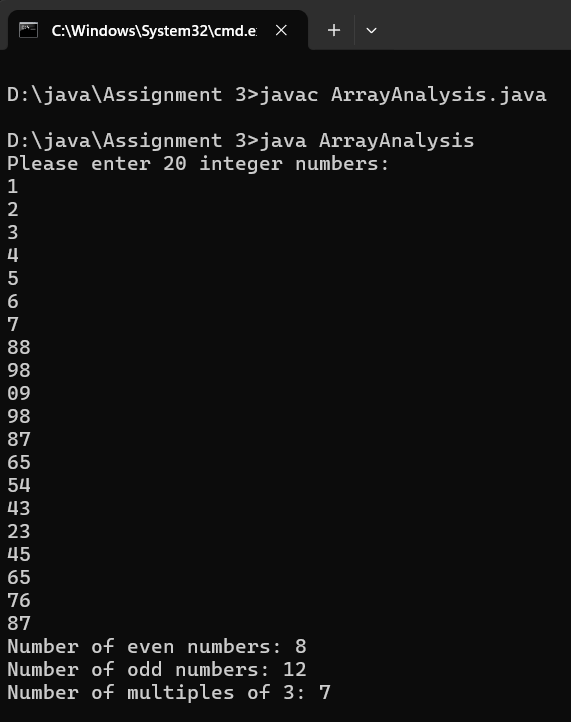
System.out.println("Number of odd numbers: " + oddCount);

System.out.println("Number of multiples of 3: " + multipleOfThreeCount);

scanner.close();

}

}



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 StudentPerformance {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int totalStudents = 12;

int[] physicsMarks = new int[totalStudents];

int[] chemistryMarks = new int[totalStudents];

int[] mathsMarks = new int[totalStudents];

int count75AndAbove = 0;

int count40AndBelow = 0;

// Input marks for 20 students

System.out.println("Enter marks for " + totalStudents + " students:");

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

System.out.println("Student " + (i + 1) + ":");

System.out.print("Physics marks: ");

physicsMarks[i] = scanner.nextInt();

System.out.print("Chemistry marks: ");

chemistryMarks[i] = scanner.nextInt();

System.out.print("Mathematics marks: ");

mathsMarks[i] = scanner.nextInt();

}

// Calculate aggregates and count students based on their performance

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

int totalMarks = physicsMarks[i] + chemistryMarks[i] + mathsMarks[i];

double percentage = (totalMarks / 300.0) \* 100;

if (percentage >= 75) {

count75AndAbove++;

}

if (percentage <= 40) {

count40AndBelow++;

}

}

// Display the results

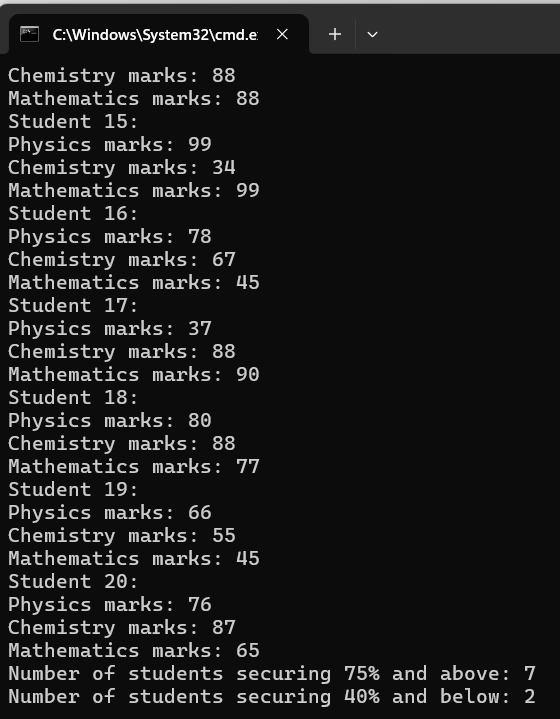
System.out.println("Number of students securing 75% and above: " + count75AndAbove);

System.out.println("Number of students securing 40% and below: " + count40AndBelow);

scanner.close();

}

}



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]

import java.util.Arrays;

public class MergeArrays {

public static void main(String[] args) {

// Initialize arrays A and B

int[] A = {1, 5, 6, 7, 8, 10};

int[] B = {2, 4, 9};

// Merge arrays A and B

int[] mergedArray = mergeArrays(A, B);

// Redistribute elements back to A and B

int p = A.length; // Number of elements A should have

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

A[i] = mergedArray[i];

}

for (int i = 0; i < mergedArray.length - p; i++) {

B[i] = mergedArray[p + i];

}

// Display the sorted arrays

System.out.println("Sorted Arrays:");

System.out.println("A: " + Arrays.toString(A));

System.out.println("B: " + Arrays.toString(B));

}

// Method to merge two sorted arrays

public static int[] mergeArrays(int[] A, int[] B) {

int p = A.length;

int q = B.length;

int[] mergedArray = new int[p + q];

int i = 0, j = 0, k = 0;

// Merge arrays A and B

while (i < p && j < q) {

if (A[i] <= B[j]) {

mergedArray[k++] = A[i++];

} else {

mergedArray[k++] = B[j++];

}

}

// Copy remaining elements of A, if any

while (i < p) {

mergedArray[k++] = A[i++];

}

// Copy remaining elements of B, if any

while (j < q) {

mergedArray[k++] = B[j++];

}

return mergedArray;

}

}

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]

import java.util.Arrays;

public class MergeArrays {

public static void main(String[] args) {

// Initialize arrays A and B

int[] A = {1, 5, 6, 7, 8, 10};

int[] B = {2, 4, 9};

// Merge arrays A and B

int[] mergedArray = mergeArrays(A, B);

// Redistribute elements back to A and B

int p = A.length; // Number of elements A should have

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

A[i] = mergedArray[i];

}

for (int i = 0; i < mergedArray.length - p; i++) {

B[i] = mergedArray[p + i];

}

// Display the sorted arrays

System.out.println("Sorted Arrays:");

System.out.println("A: " + Arrays.toString(A));

System.out.println("B: " + Arrays.toString(B));

}

// Method to merge two sorted arrays

public static int[] mergeArrays(int[] A, int[] B) {

int p = A.length;

int q = B.length;

int[] mergedArray = new int[p + q];

int i = 0, j = 0, k = 0;

// Merge arrays A and B

while (i < p && j < q) {

if (A[i] <= B[j]) {

mergedArray[k++] = A[i++];

} else {

mergedArray[k++] = B[j++];

}

}

// Copy remaining elements of A, if any

while (i < p) {

mergedArray[k++] = A[i++];

}

// Copy remaining elements of B, if any

while (j < q) {

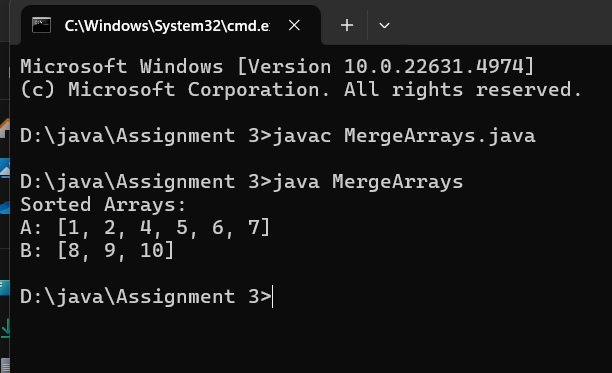
mergedArray[k++] = B[j++];

}

return mergedArray;

}

}



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

public class MaxProduct {

public static void main(String[] args) {

int[] nums = {2, 3, 5, 7, -7, 5, 8, -5};

findMaxProduct(nums);

}

public static void findMaxProduct(int[] nums) {

if (nums == null || nums.length < 2) {

System.out.println("Array should have at least two elements.");

return;

}

// Initialize variables to store the two largest and two smallest numbers

int max1 = Integer.MIN\_VALUE, max2 = Integer.MIN\_VALUE;

int min1 = Integer.MAX\_VALUE, min2 = Integer.MAX\_VALUE;

// Traverse the array to find the required numbers

for (int num : nums) {

if (num > max1) {

max2 = max1;

max1 = num;

} else if (num > max2) {

max2 = num;

}

if (num < min1) {

min2 = min1;

min1 = num;

} else if (num < min2) {

min2 = num;

}

}

// Calculate the products of the two largest and two smallest numbers

int product1 = max1 \* max2;

int product2 = min1 \* min2;

// Determine the maximum product

if (product1 > product2) {

System.out.println("Pair is (" + max1 + ", " + max2 + "), Maximum Product: " + product1);

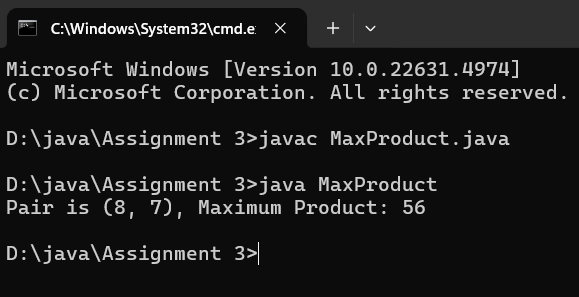
} else {

System.out.println("Pair is (" + min1 + ", " + min2 + "), Maximum Product: " + product2);

}

}

}



18. Print a Matrix

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

public class MatrixPrinter {

public static void main(String[] args) {

// Example m x n matrix

int[][] matrix = {

{1, 2, 3},

{4, 5, 6},

{7, 8, 9}

};

// Iterate through each row

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

// Iterate through each column in the current row

for (int j = 0; j < matrix[i].length; j++) {

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

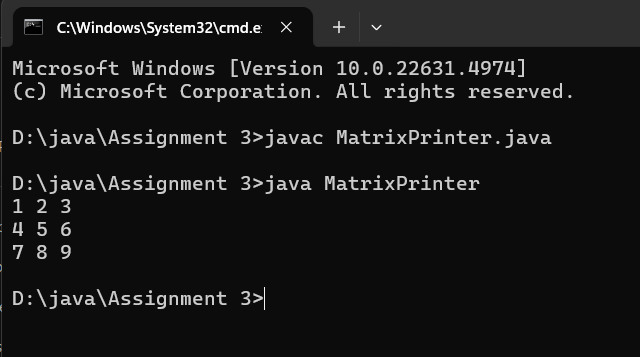
}

System.out.println(); // Move to the next line after printing a row

}

}

}



19. Transpose of a Matrix

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

public class MatrixTranspose {

public static void main(String[] args) {

// Define the original matrix

int[][] originalMatrix = {

{1, 2, 3},

{4, 5, 6},

{7, 8, 9}

};

// Calculate the dimensions of the original matrix

int rows = originalMatrix.length;

int cols = originalMatrix[0].length;

// Initialize the transposed matrix

int[][] transposedMatrix = new int[cols][rows];

// Populate the transposed matrix

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

for (int j = 0; j < cols; j++) {

transposedMatrix[j][i] = originalMatrix[i][j];

}

}

// Display the original matrix

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

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

for (int j = 0; j < cols; j++) {

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

}

System.out.println();

}

// Display the transposed matrix

System.out.println("\nTransposed Matrix:");

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

for (int j = 0; j < rows; j++) {

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

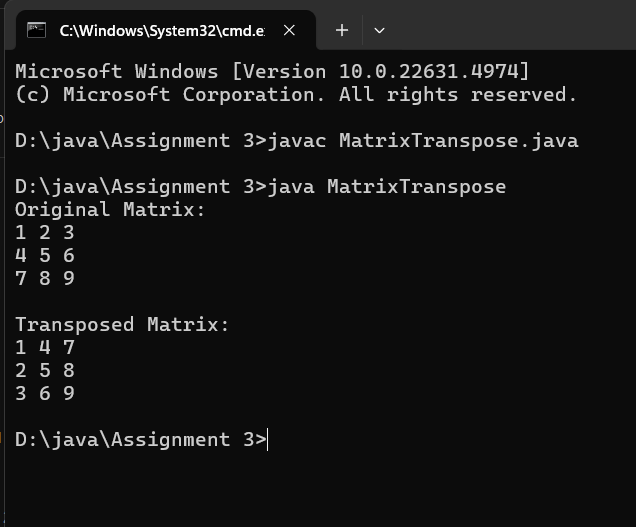
}

System.out.println();

}

}

}



Multiply two matrices and return the resultant matrix.

public class MatrixMultiplication {

public static void main(String[] args) {

// Define the first matrix (2x3)

int[][] matrixA = {

{1, 2, 3},

{4, 5, 6}

};

// Define the second matrix (3x2)

int[][] matrixB = {

{7, 8},

{9, 10},

{11, 12}

};

// Check if multiplication is possible

if (matrixA[0].length != matrixB.length) {

System.out.println("Matrix multiplication is not possible. The number of columns in matrix A must equal the number of rows in matrix B.");

return;

}

// Initialize the resultant matrix (2x2)

int[][] resultMatrix = new int[matrixA.length][matrixB[0].length];

// Perform matrix multiplication

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

for (int j = 0; j < matrixB[0].length; j++) {

for (int k = 0; k < matrixB.length; k++) {

resultMatrix[i][j] += matrixA[i][k] \* matrixB[k][j];

}

}

}

// Display the resultant matrix

System.out.println("Resultant Matrix:");

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

for (int j = 0; j < resultMatrix[i].length; j++) {

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

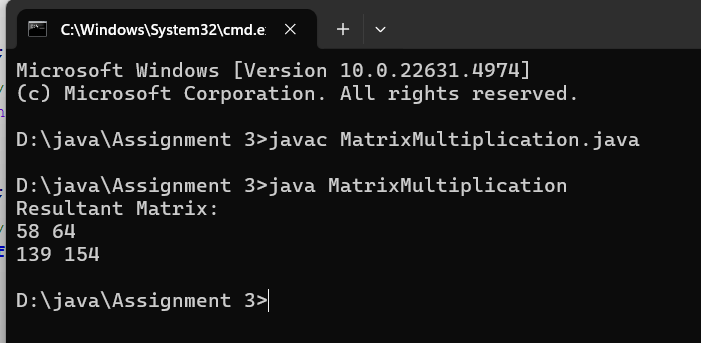
}

System.out.println();

}

}

}



24. Rotate a Matrix by 90 Degrees

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

public class MatrixRotation {

public static void main(String[] args) {

// Define the original matrix

int[][] matrix = {

{1, 2, 3},

{4, 5, 6},

{7, 8, 9}

};

// Rotate the matrix by 90 degrees clockwise

rotateMatrixBy90Degrees(matrix);

// Display the rotated matrix

System.out.println("Rotated Matrix:");

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

for (int j = 0; j < matrix[i].length; j++) {

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

}

System.out.println();

}

}

// Function to rotate the matrix by 90 degrees clockwise

public static void rotateMatrixBy90Degrees(int[][] matrix) {

int n = matrix.length;

// Transpose the matrix

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

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

int temp = matrix[i][j];

matrix[i][j] = matrix[j][i];

matrix[j][i] = temp;

}

}

// Reverse each row

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

int start = 0;

int end = n - 1;

while (start < end) {

int temp = matrix[i][start];

matrix[i][start] = matrix[i][end];

matrix[i][end] = temp;

start++;

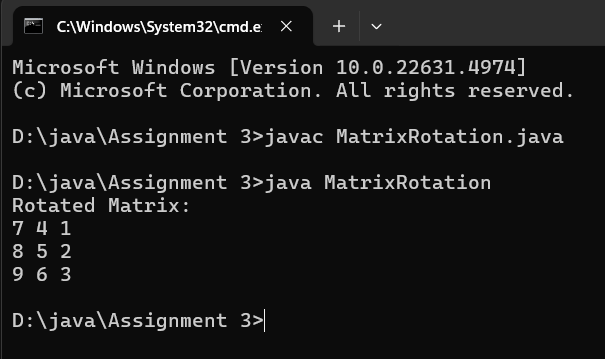
end--;

}

}

}

}



25. Find the Diagonal Sum

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

public class DiagonalSum {

public static void main(String[] args) {

// Define a square matrix

int[][] matrix = {

{1, 2, 3},

{4, 5, 6},

{7, 8, 9}

};

int n = matrix.length;

int primaryDiagonalSum = 0;

int secondaryDiagonalSum = 0;

// Iterate through the matrix

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

primaryDiagonalSum += matrix[i][i]; // Primary diagonal

secondaryDiagonalSum += matrix[i][n - 1 - i]; // Secondary diagonal

}

// If the matrix has an odd dimension, subtract the center element

if (n % 2 != 0) {

int centerValue = matrix[n / 2][n / 2];

primaryDiagonalSum -= centerValue;

secondaryDiagonalSum -= centerValue;

}

// Display the results

System.out.println("Primary Diagonal Sum: " + primaryDiagonalSum);

System.out.println("Secondary Diagonal Sum: " + secondaryDiagonalSum);

System.out.println("Total Sum (Primary + Secondary - Center): " +

(primaryDiagonalSum + secondaryDiagonalSum));

}

}

