Array coding question :

1. Find the Largest and Smallest Element

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

Ans : public class MinMaxFinder {

public static void findLargestAndSmallest(int[] arr) {

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

System.out.println("Array is empty.");

return;

}

int smallest = arr[0];

int largest = arr[0];

for (int num : arr) {

if (num < smallest) {

smallest = num;

} else if (num > largest) {

largest = num;

}

}

System.out.println("Smallest: " + smallest);

System.out.println("Largest: " + largest);

}

public static void main(String[] args) {

int[] arr = {12, 5, 8, 19, 3, 25, 7};

findLargestAndSmallest(arr);

}

}

2. Reverse an Array

○ Reverse the given array in place.

Ans : import java.util.Arrays;

public class ReverseArray {

public static void reverse(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, 6};

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

reverse(arr);

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

}

}

3. Find the Second Largest Element

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

Ans : public class SecondLargest {

public static int findSecondLargest(int[] arr) {

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

throw new IllegalArgumentException("Array must have at least two elements.");

}

int largest = Integer.MIN\_VALUE;

int secondLargest = Integer.MIN\_VALUE;

for (int num : arr) {

if (num > largest) {

secondLargest = largest;

largest = num;

} else if (num > secondLargest && num < largest) {

secondLargest = num;

}

}

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

throw new RuntimeException("No second largest element found.");

}

return secondLargest;

}

public static void main(String[] args) {

int[] arr = {12, 35, 1, 10, 34, 1};

System.out.println("Second Largest: " + findSecondLargest(arr));

}

}

4. Count Even and Odd Numbers

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

Ans : public class CountEvenOdd {

public static void countEvenOdd(int[] arr) {

int evenCount = 0, oddCount = 0;

for (int num : arr) {

if (num % 2 == 0) {

evenCount++;

} else {

oddCount++;

}

}

System.out.println("Even Count: " + evenCount);

System.out.println("Odd Count: " + oddCount);

}

public static void main(String[] args) {

int[] arr = {12, 7, 5, 8, 10, 3, 14, 21};

countEvenOdd(arr);

}

}

5. Find Sum and Average

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

Ans : public class SumAndAverage {

public static void calculateSumAndAverage(int[] arr) {

if (arr.length == 0) {

System.out.println("Array is empty.");

return;

}

int sum = 0;

for (int num : arr) {

sum += num;

}

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

System.out.println("Sum: " + sum);

System.out.println("Average: " + average);

}

public static void main(String[] args) {

int[] arr = {12, 7, 5, 8, 10, 3, 14, 21};

calculateSumAndAverage(arr);

}

}

6. Remove Duplicates from a Sorted Array

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

Ans : public class RemoveDuplicates {

public static int removeDuplicates(int[] arr) {

if (arr.length == 0) return 0;

int uniqueIndex = 0; // Pointer for unique elements

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

if (arr[i] != arr[uniqueIndex]) {

uniqueIndex++;

arr[uniqueIndex] = arr[i];

}

}

return uniqueIndex + 1; // New length of unique elements

}

public static void main(String[] args) {

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

int newLength = removeDuplicates(arr);

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

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

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

}

}

}

7. Rotate an Array

○ Rotate the array to the right by k positions.

Ans : import java.util.Arrays;

public class RotateArray {

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

int n = arr.length;

k = k % n; // Handle cases where k > n

// Step 1: Reverse the entire array

reverse(arr, 0, n - 1);

// Step 2: Reverse the first k elements

reverse(arr, 0, k - 1);

// Step 3: Reverse the remaining n-k elements

reverse(arr, k, n - 1);

}

private static void reverse(int[] arr, int left, int right) {

while (left < right) {

int temp = arr[left];

arr[left] = arr[right];

arr[right] = temp;

left++;

right--;

}

}

public static void main(String[] args) {

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

int k = 3; // Rotate right by 3 positions

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

rotate(arr, k);

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

}

}

8. Merge Two Sorted Arrays

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

Ans : import java.util.Arrays;

public class MergeSortedArrays {

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

int n = arr1.length, m = arr2.length;

int totalLength = n + m;

int gap = (totalLength + 1) / 2; // Initial gap

while (gap > 0) {

int i = 0, j = gap;

while (j < totalLength) {

if (j < n && arr1[i] > arr1[j]) {

swap(arr1, i, j);

} else if (i < n && j >= n && arr1[i] > arr2[j - n]) {

swap(arr1, arr2, i, j - n);

} else if (i >= n && arr2[i - n] > arr2[j - n]) {

swap(arr2, i - n, j - n);

}

i++;

j++;

}

gap = (gap > 1) ? (gap + 1) / 2 : 0;

}

}

private static void swap(int[] arr, int i, int j) {

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

private static void swap(int[] arr1, int[] arr2, int i, int j) {

int temp = arr1[i];

arr1[i] = arr2[j];

arr2[j] = temp;

}

public static void main(String[] args) {

int[] arr1 = {1, 4, 7, 8, 10};

int[] arr2 = {2, 3, 9};

merge(arr1, arr2);

System.out.println("Merged Array 1: " + Arrays.toString(arr1));

System.out.println("Merged Array 2: " + Arrays.toString(arr2));

}

}

9. Find Missing Number in an Array

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

Ans : public class MissingNumber {

public static int findMissingNumber(int[] arr, int n) {

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

int actualSum = 0;

for (int num : arr) {

actualSum += num;

}

return expectedSum - actualSum; // Missing number

}

public static void main(String[] args) {

int[] arr = {1, 2, 4, 5, 6}; // Missing number is 3

int n = 6; // Since array size is n-1, the full range is 1 to 6

System.out.println("Missing Number: " + findMissingNumber(arr, n));

}

}

10. Find Intersection and Union of Two Arrays

○ Find the intersection and union of two unsorted arrays.

Ans : import java.util.\*;

public class UnionIntersection {

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

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

for (int num : arr1) {

unionSet.add(num);

}

for (int num : arr2) {

unionSet.add(num);

}

System.out.println("Union: " + unionSet);

}

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

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

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

for (int num : arr1) {

set1.add(num);

}

for (int num : arr2) {

if (set1.contains(num)) {

intersectionSet.add(num);

}

}

System.out.println("Intersection: " + intersectionSet);

}

public static void main(String[] args) {

int[] arr1 = {1, 3, 4, 5, 7};

int[] arr2 = {2, 3, 5, 6};

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.

Ans : public class SubarrayWithSum {

public static void findSubarrayWithSum(int[] arr, int S) {

int left = 0, sum = 0;

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

sum += arr[right]; // Expand window

while (sum > S && left <= right) {

sum -= arr[left]; // Shrink window

left++;

}

if (sum == S) {

System.out.println("Subarray found from index " + left + " to " + right);

return;

}

}

System.out.println("No subarray found with sum " + S);

}

public static void main(String[] args) {

int[] arr = {1, 4, 20, 3, 10, 5};

int S = 33;

findSubarrayWithSum(arr, 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

Ans : import java.util.Scanner;

public class ArrayAnalysis {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int[] arr = new int[20];

// Input 20 numbers

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

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

arr[i] = scanner.nextInt();

}

int evenCount = 0, oddCount = 0, multipleOf3Count = 0;

// Count Even, Odd, Multiples of 3

for (int num : arr) {

if (num % 2 == 0) evenCount++;

else oddCount++;

if (num % 3 == 0) multipleOf3Count++;

}

// Display 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: " + multipleOf3Count);

scanner.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.

Ans : import java.util.Scanner;

public class StudentMarksAnalysis {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int numStudents = 20;

int[] marks = new int[numStudents]; // Array to store total marks

// Input marks for 20 students

System.out.println("Enter marks for 20 students (Physics, Chemistry, Maths):");

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

System.out.print("Student " + (i + 1) + " - Physics: ");

int physics = scanner.nextInt();

System.out.print("Student " + (i + 1) + " - Chemistry: ");

int chemistry = scanner.nextInt();

System.out.print("Student " + (i + 1) + " - Maths: ");

int maths = scanner.nextInt();

marks[i] = physics + chemistry + maths; // Store total marks

}

int countAbove75 = 0, countBelow40 = 0;

int totalMarks = 300; // Maximum marks (100 each subject)

// Count students based on aggregate percentage

for (int total : marks) {

double percentage = (total \* 100.0) / totalMarks;

if (percentage >= 75) countAbove75++;

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);

scanner.close();

}

}

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

Ans : import java.util.ArrayList;

import java.util.Scanner;

public class EvenOddSeparation {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int[] arr = new int[20]; // Main array for 20 numbers

ArrayList<Integer> even = new ArrayList<>();

ArrayList<Integer> odd = new ArrayList<>();

// Accept 20 numbers

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

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

arr[i] = scanner.nextInt();

// Separate even and odd numbers

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

even.add(arr[i]);

} else {

odd.add(arr[i]);

}

}

// Print even numbers

System.out.println("Even numbers: " + even);

// Print odd numbers

System.out.println("Odd numbers: " + odd);

scanner.close();

}

}

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]

Ans : import java.util.\*;

public class ZeroSumSubarrays {

public static void findZeroSumSubarrays(int[] nums) {

System.out.println("Sub-arrays with 0 sum:");

Map<Integer, List<Integer>> map = new HashMap<>();

int sum = 0;

// Initialize map with sum 0 at index -1 for subarrays starting from index 0

map.put(0, new ArrayList<>());

map.get(0).add(-1);

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

sum += nums[i];

if (map.containsKey(sum)) {

for (int startIndex : map.get(sum)) {

System.out.println(Arrays.toString(Arrays.copyOfRange(nums, startIndex + 1, i + 1)));

}

}

// Store index in map for future subarray calculations

map.putIfAbsent(sum, new ArrayList<>());

map.get(sum).add(i);

}

}

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};

findZeroSumSubarrays(nums1);

findZeroSumSubarrays(nums2);

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]

Ans : import java.util.Arrays;

public class MergeSortedArrays {

public static void mergeSortedArrays(int[] A, int[] B) {

int p = A.length, q = B.length;

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

// Copy both arrays into merged[]

System.arraycopy(A, 0, merged, 0, p);

System.arraycopy(B, 0, merged, p, q);

// Sort the merged array

Arrays.sort(merged);

// Fill A with first p smallest elements

System.arraycopy(merged, 0, A, 0, p);

// Fill B with remaining elements

System.arraycopy(merged, p, B, 0, q);

// Print results

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};

mergeSortedArrays(A, B);

}

}

**Output**

Sorted Arrays:

A: [1, 2, 4, 5, 6, 7]

B: [8, 9, 10]

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

Ans : import java.util.Arrays;

public class MaxProductPair {

public static void findMaxProductPair(int[] nums) {

if (nums.length < 2) {

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

return;

}

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

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

// Find the two largest and two smallest 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;

}

}

// Compare max product of two largest vs. two smallest numbers

int product1 = max1 \* max2;

int product2 = min1 \* min2;

if (product1 > product2) {

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

} else {

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

}

}

public static void main(String[] args) {

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

findMaxProductPair(nums);

}

}

**Output**

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

18. Print a Matrix

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

Ans : import java.util.Scanner;

public class MatrixRowWisePrint {

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

int rows = matrix.length;

int cols = matrix[0].length;

System.out.println("Matrix elements row-wise:");

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

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

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

}

System.out.println(); // Move to the next row

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input matrix dimensions

System.out.print("Enter number of rows (m): ");

int m = scanner.nextInt();

System.out.print("Enter number of columns (n): ");

int n = scanner.nextInt();

int[][] matrix = new int[m][n];

// Input matrix elements

System.out.println("Enter the matrix elements row-wise:");

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

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

matrix[i][j] = scanner.nextInt();

}

}

// Print the matrix

printMatrix(matrix);

scanner.close();

}

}

**Example** :

Enter number of rows (m): 2

Enter number of columns (n): 3

Enter the matrix elements row-wise:

1 2 3

4 5 6

Matrix elements row-wise:

1 2 3

4 5 6

19. Transpose of a Matrix ○ Given a matrix, return its transpose (swap rows and columns).

Ans : import java.util.Scanner;

public class MatrixTranspose {

public static int[][] transposeMatrix(int[][] matrix, int rows, int cols) {

int[][] transposed = new int[cols][rows]; // New matrix with swapped dimensions

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

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

transposed[j][i] = matrix[i][j]; // Swap rows and columns

}

}

return transposed;

}

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

for (int[] row : matrix) {

for (int elem : row) {

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

}

System.out.println();

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter number of rows: ");

int m = scanner.nextInt();

System.out.print("Enter number of columns: ");

int n = scanner.nextInt();

int[][] matrix = new int[m][n];

System.out.println("Enter matrix elements:");

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

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

matrix[i][j] = scanner.nextInt();

}

}

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

printMatrix(transposeMatrix(matrix, m, n));

scanner.close();

}

}

**Example Run**

Enter number of rows: 2

Enter number of columns: 3

Enter matrix elements:

1 2 3

4 5 6

Transposed Matrix:

1 4

2 5

3 6

20. Sum of Two Matrices Given two matrices of the same size, compute their sum

Ans : import java.util.Scanner;

public class MatrixSum {

public static int[][] addMatrices(int[][] matrix1, int[][] matrix2, int rows, int cols) {

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

// Compute sum of corresponding elements

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

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

sumMatrix[i][j] = matrix1[i][j] + matrix2[i][j];

}

}

return sumMatrix;

}

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

for (int[] row : matrix) {

for (int elem : row) {

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

}

System.out.println();

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input matrix dimensions

System.out.print("Enter number of rows: ");

int rows = scanner.nextInt();

System.out.print("Enter number of columns: ");

int cols = scanner.nextInt();

int[][] matrix1 = new int[rows][cols];

int[][] matrix2 = new int[rows][cols];

// Input first matrix

System.out.println("Enter first matrix elements:");

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

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

matrix1[i][j] = scanner.nextInt();

}

}

// Input second matrix

System.out.println("Enter second matrix elements:");

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

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

matrix2[i][j] = scanner.nextInt();

}

}

// Compute and print the sum matrix

int[][] sumMatrix = addMatrices(matrix1, matrix2, rows, cols);

System.out.println("\nSum of Matrices:");

printMatrix(sumMatrix);

scanner.close();

}

}

**Example Run**

Enter number of rows: 2

Enter number of columns: 3

Enter first matrix elements:

1 2 3

4 5 6

Enter second matrix elements:

7 8 9

1 2 3

Sum of Matrices:

8 10 12

5 7 9

21. Row-wise and Column-wise Sum

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

Ans : import java.util.Scanner;

public class MatrixRowColumnSum {

public static void computeRowColumnSum(int[][] matrix, int rows, int cols) {

// Compute row-wise sum

System.out.println("\nRow-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("Sum of row " + (i + 1) + ": " + rowSum);

}

// Compute column-wise sum

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("Sum of column " + (j + 1) + ": " + colSum);

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input matrix dimensions

System.out.print("Enter number of rows: ");

int rows = scanner.nextInt();

System.out.print("Enter number of columns: ");

int cols = scanner.nextInt();

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

// Input matrix elements

System.out.println("Enter the matrix elements:");

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

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

matrix[i][j] = scanner.nextInt();

}

}

// Compute and display row-wise and column-wise sums

computeRowColumnSum(matrix, rows, cols);

scanner.close();

}

}

**Example Run**

Enter number of rows: 3

Enter number of columns: 3

Enter the matrix elements:

1 2 3

4 5 6

7 8 9

Row-wise Sum:

Sum of row 1: 6

Sum of row 2: 15

Sum of row 3: 24

Column-wise Sum:

Sum of column 1: 12

Sum of column 2: 15

Sum of column 3: 18

22. Find the Maximum Element in a Matrix

* Find the largest element in a given matrix

Ans : import java.util.Scanner;

public class MatrixMaxElement {

public static int findMaxElement(int[][] matrix, int rows, int cols) {

int maxElement = Integer.MIN\_VALUE; // Initialize with the smallest possible value

// Traverse the matrix to find the maximum element

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

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

if (matrix[i][j] > maxElement) {

maxElement = matrix[i][j];

}

}

}

return maxElement;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input matrix dimensions

System.out.print("Enter number of rows: ");

int rows = scanner.nextInt();

System.out.print("Enter number of columns: ");

int cols = scanner.nextInt();

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

// Input matrix elements

System.out.println("Enter the matrix elements:");

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

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

matrix[i][j] = scanner.nextInt();

}

}

// Find and display the maximum element

int maxElement = findMaxElement(matrix, rows, cols);

System.out.println("\nThe largest element in the matrix is: " + maxElement);

scanner.close();

}

}

**Example** **Run**

Enter number of rows: 3

Enter number of columns: 3

Enter the matrix elements:

1 2 3

4 9 6

7 8 5

The largest element in the matrix is: 9

23 . Matrix Multiplication

○ Multiply two matrices and return the resultant matrix.

Ans : import java.util.Scanner;

public class MatrixMultiplication {

public static int[][] multiplyMatrices(int[][] matrix1, int[][] matrix2, int rows1, int cols1, int cols2) {

int[][] result = new int[rows1][cols2];

// Matrix multiplication logic

for (int i = 0; i < rows1; i++) {

for (int j = 0; j < cols2; j++) {

for (int k = 0; k < cols1; k++) {

result[i][j] += matrix1[i][k] \* matrix2[k][j];

}

}

}

return result;

}

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

for (int[] row : matrix) {

for (int elem : row) {

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

}

System.out.println();

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input dimensions for first matrix

System.out.print("Enter number of rows for first matrix: ");

int rows1 = scanner.nextInt();

System.out.print("Enter number of columns for first matrix (must match rows of second matrix): ");

int cols1 = scanner.nextInt();

// Input dimensions for second matrix

System.out.print("Enter number of columns for second matrix: ");

int cols2 = scanner.nextInt();

int[][] matrix1 = new int[rows1][cols1];

int[][] matrix2 = new int[cols1][cols2];

// Input first matrix

System.out.println("Enter first matrix elements:");

for (int i = 0; i < rows1; i++) {

for (int j = 0; j < cols1; j++) {

matrix1[i][j] = scanner.nextInt();

}

}

// Input second matrix

System.out.println("Enter second matrix elements:");

for (int i = 0; i < cols1; i++) {

for (int j = 0; j < cols2; j++) {

matrix2[i][j] = scanner.nextInt();

}

}

// Compute matrix multiplication

int[][] resultMatrix = multiplyMatrices(matrix1, matrix2, rows1, cols1, cols2);

// Print result

System.out.println("\nResultant Matrix after Multiplication:");

printMatrix(resultMatrix);

scanner.close();

}

}

**Example Run**

Enter number of rows for first matrix: 2

Enter number of columns for first matrix (must match rows of second matrix): 3

Enter number of columns for second matrix: 2

Enter first matrix elements:

1 2 3

4 5 6

Enter second matrix elements:

7 8

9 10

11 12

Resultant Matrix after Multiplication:

58 64

139 154

24. Rotate a Matrix by 90 Degrees

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

Ans : import java.util.Scanner;

public class RotateMatrix90 {

public static void rotateMatrix(int[][] matrix, int N) {

// Step 1: Transpose the matrix

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

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

for (int j = 0; j < N / 2; j++) {

int temp = matrix[i][j];

matrix[i][j] = matrix[i][N - j - 1];

matrix[i][N - j - 1] = temp;

}

}

}

public static void printMatrix(int[][] matrix, int N) {

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

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

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

}

System.out.println();

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input matrix size

System.out.print("Enter matrix size (N x N): ");

int N = scanner.nextInt();

int[][] matrix = new int[N][N];

// Input matrix elements

System.out.println("Enter the matrix elements:");

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

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

matrix[i][j] = scanner.nextInt();

}

}

// Rotate matrix by 90 degrees

rotateMatrix(matrix, N);

// Print the rotated matrix

System.out.println("\nRotated Matrix (90 degrees clockwise):");

printMatrix(matrix, N);

scanner.close();

}

}

**Example Run**

Enter matrix size (N x N): 3

Enter the matrix elements:

1 2 3

4 5 6

7 8 9

Rotated Matrix (90 degrees clockwise):

7 4 1

8 5 2

9 6 3

25. Find the Diagonal Sum

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

Ans : import java.util.Scanner;

public class DiagonalSum {

public static int findDiagonalSum(int[][] matrix, int N) {

int sum = 0;

// Sum both primary and secondary diagonals

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

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

if (i != N - i - 1) { // Avoid double-counting center in odd-sized matrices

sum += matrix[i][N - i - 1]; // Secondary diagonal

}

}

return sum;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input matrix size

System.out.print("Enter matrix size (N x N): ");

int N = scanner.nextInt();

int[][] matrix = new int[N][N];

// Input matrix elements

System.out.println("Enter the matrix elements:");

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

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

matrix[i][j] = scanner.nextInt();

}

}

// Compute and display diagonal sum

int diagonalSum = findDiagonalSum(matrix, N);

System.out.println("\nSum of both diagonals: " + diagonalSum);

scanner.close();

}

}

**Example Run**

Enter matrix size (N x N): 3

Enter the matrix elements:

1 2 3

4 5 6

7 8 9

Sum of both diagonals: 25