ASSIGNMENT -3

Array coding question:

```
import java.util.*;


public class ArrayProblems {


   // 1. Find Largest and Smallest Element
   public static void findLargestSmallest(int[] arr) {
      int largest = arr[0], smallest = arr[0];
      for (int num : arr) {
         if (num > largest) largest = num;
         if (num < smallest) smallest = num;
      }
      System.out.println("Smallest: " + smallest + ", Largest: " + largest);
   }
```

```
   // 2. Reverse an Array
   public static void reverseArray(int[] arr) {
      int n = arr.length;
      for (int i = 0; i < n / 2; i++) {
         int temp = arr[i];
         arr[i] = arr[n - i - 1];
         arr[n - i - 1] = temp;
      }
      System.out.println("Reversed Array: " + Arrays.toString(arr));
   }
```

```
    ------------------------------------------------------------------------------------------------
---


    // 3. Find the Second Largest Element
    public static void secondLargest(int[] arr) {
        int first = Integer.MIN_VALUE, second = Integer.MIN_VALUE;
        for (int num : arr) {
            if (num > first) {
                second = first;
                first = num;
            } else if (num > second && num != first) {
                second = num;
            }
        }
        System.out.println("Second Largest: " + (second == Integer.MIN_VALUE ?
"No second largest" : second));
    }
 ------------------------------------------------------------------------------------------------
-


    // 4. Count Even and Odd Numbers
    public static void countEvenOdd(int[] arr) {
        int even = 0, odd = 0;
        for (int num : arr) {
            if (num % 2 == 0) even++;
            else odd++;
        }
```

```
        System.out.println("Even: " + even + ", Odd: " + odd);
    }
```

```
    // 5. Find Sum and Average
    public static void sumAndAverage(int[] arr) {
        int sum = 0;
        for (int num : arr) sum += num;
        System.out.println("Sum: " + sum + ", Average: " + (double) sum /
arr.length);
    }
```

```
    // 6. Remove Duplicates from a Sorted Array
    public static void removeDuplicates(int[] arr) {
        int index = 0;
        for (int i = 1; i < arr.length; i++) {
            if (arr[i] != arr[index]) arr[++index] = arr[i];
        }
        System.out.println("Array after removing duplicates: " +
Arrays.toString(Arrays.copyOf(arr, index + 1)));
    }
```

```
    // 7. Rotate an Array by k positions
    public static void rotateArray(int[] arr, int k) {
        k %= arr.length;
```

```
        reverse(arr, 0, arr.length - 1);
        reverse(arr, 0, k - 1);
        reverse(arr, k, arr.length - 1);
        System.out.println("Rotated Array: " + Arrays.toString(arr));
    }
    private static void reverse(int[] arr, int start, int end) {
        while (start < end) {
            int temp = arr[start];
            arr[start++] = arr[end];
            arr[end--] = temp;
        }
    }
```

```
    // 8. Merge Two Sorted Arrays
    public static void mergeSortedArrays(int[] arr1, int[] arr2) {
        int i = 0, j = 0;
        List<Integer> result = new ArrayList<>();
        while (i < arr1.length && j < arr2.length) {
            if (arr1[i] < arr2[j]) result.add(arr1[i++]);
            else result.add(arr2[j++]);
        }
        while (i < arr1.length) result.add(arr1[i++]);
        while (j < arr2.length) result.add(arr2[j++]);
        System.out.println("Merged Sorted Array: " + result);
    }
```

```
    // 9. Find Missing Number in an Array
    public static void findMissingNumber(int[] arr, int n) {
        int total = n * (n + 1) / 2;
        int sum = 0;
        for (int num : arr) sum += num;
        System.out.println("Missing Number: " + (total - sum));
    }
```

```
    // 10. Find Intersection and Union of Two Arrays
    public static void findIntersectionUnion(int[] arr1, int[] arr2) {
        Set<Integer> union = new HashSet<>(), intersection = new HashSet<>();
        for (int num : arr1) union.add(num);
        for (int num : arr2) {
            if (union.contains(num)) intersection.add(num);
            union.add(num);
        }
        System.out.println("Union: " + union);
        System.out.println("Intersection: " + intersection);
    }
```

--

```
    // 11. Find a Subarray with Given Sum
    public static void findSubarrayWithSum(int[] arr, int S) {
```

```
        for (int i = 0; i < arr.length; i++) {
            int sum = 0;
            for (int j = i; j < arr.length; j++) {
                sum += arr[j];
                if (sum == S) {
                    System.out.println("Subarray with sum " + S + " found: " +
Arrays.toString(Arrays.copyOfRange(arr, i, j + 1)));
                }
            }
        }
    }
```

```
    // 15. Find all subarrays with 0 sum
    public static void findZeroSumSubarrays(int[] arr) {
        Map<Integer, List<Integer>> map = new HashMap<>();
        int sum = 0;
        for (int i = 0; i < arr.length; i++) {
            sum += arr[i];
            if (sum == 0) System.out.println("Subarray with 0 sum: " +
Arrays.toString(Arrays.copyOfRange(arr, 0, i + 1)));
            if (map.containsKey(sum)) {
                for (int start : map.get(sum)) {
                    System.out.println("Subarray with 0 sum: " +
Arrays.toString(Arrays.copyOfRange(arr, start + 1, i + 1)));
                }
            }
```

```
            map.putIfAbsent(sum, new ArrayList<>());
            map.get(sum).add(i);
        }
    }

    public static void main(String[] args) {
        int[] arr = {1, 3, -7, 3, 2, 3, 1, -3, -2, -2};
        findLargestSmallest(arr);
        reverseArray(arr);
        secondLargest(arr);
        countEvenOdd(arr);
        sumAndAverage(arr);
        rotateArray(arr, 2);
        findZeroSumSubarrays(arr);
    }

import java.util.*;

public class ArrayProblems
```

```
    // 12. Count Even, Odd, and Multiples of 3
    public static void countNumbers(int[] arr) {
        int even = 0, odd = 0, multiplesOf3 = 0;
        for (int num : arr) {
            if (num % 2 == 0) even++;
```

```
            else odd++;
            if (num % 3 == 0) multiplesOf3++;
        }
        System.out.println("Even: " + even + ", Odd: " + odd + ", Multiples of 3: " +
multiplesOf3);
    }
```

```
    // 13. Student Marks Analysis
    public static void studentMarksAnalysis(int[][] marks) {
        int above75 = 0, below40 = 0;
        for (int[] student : marks) {
            int total = student[0] + student[1] + student[2];
            double percentage = (double) total / 3;
            if (percentage >= 75) above75++;
            if (percentage <= 40) below40++;
        }
        System.out.println("Students with 75% and above: " + above75);
        System.out.println("Students with 40% and below: " + below40);
    }
```

```
    // 14. Separate Even and Odd Numbers
    public static void separateEvenOdd(int[] arr) {
        List<Integer> even = new ArrayList<>(), odd = new ArrayList<>();
        for (int num : arr) {
            if (num % 2 == 0) even.add(num);
```

```
            else odd.add(num);
        }
        System.out.println("Even numbers: " + even);
        System.out.println("Odd numbers: " + odd);
    }

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

        // Input for question 12
        int[] numbers = new int[20];
        System.out.println("Enter 20 numbers:");
        for (int i = 0; i < 20; i++) numbers[i] = scanner.nextInt();
        countNumbers(numbers);
        separateEvenOdd(numbers);

        // Input for question 13
        int[][] marks = new int[20][3];
        System.out.println("Enter marks for 20 students (Physics, Chemistry,
Maths):");
        for (int i = 0; i < 20; i++) {
            for (int j = 0; j < 3; j++) {
                marks[i][j] = scanner.nextInt();
            }
        }
        studentMarksAnalysis(marks);
    }
```

```
}


public class MatrixOperations {
  ------------------------------------------------------------------------------------------------------
--

   // 16. Merge Two Sorted Arrays
   public static void mergeSortedArrays(int[] A, int[] B) {
      int p = A.length, q = B.length;
      int[] merged = new int[p + q];
      System.arraycopy(A, 0, merged, 0, p);
      System.arraycopy(B, 0, merged, p, q);
      Arrays.sort(merged);

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

      System.out.println("A: " + Arrays.toString(A));
      System.out.println("B: " + Arrays.toString(B));
   }
   ------------------------------------------------------------------------------------------------------
---
   // 17. Maximum Product Pair
   public static void maxProductPair(int[] nums) {
      Arrays.sort(nums);
      int n = nums.length;
```

```
        int prod1 = nums[0] * nums[1];
        int prod2 = nums[n - 1] * nums[n - 2];
        if (prod1 > prod2)
            System.out.println("Pair: (" + nums[0] + ", " + nums[1] + "), Max Product: " + prod1);
        else
            System.out.println("Pair: (" + nums[n - 2] + ", " + nums[n - 1] + "), Max Product: " + prod2);
    }
```

```
    // 18. Print Matrix
    public static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            System.out.println(Arrays.toString(row));
        }
    }
```

```
    // 19. Transpose Matrix
    public static int[][] transposeMatrix(int[][] matrix) {
        int m = matrix.length, n = matrix[0].length;
        int[][] transposed = new int[n][m];
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                transposed[j][i] = matrix[i][j];
```

```
        }
    }
    return transposed;
}
```

```
// 20. Sum of Two Matrices
public static int[][] sumMatrices(int[][] A, int[][] B) {
    int m = A.length, n = A[0].length;
    int[][] sum = new int[m][n];
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            sum[i][j] = A[i][j] + B[i][j];
        }
    }
    return sum;
}
```

```
// 21. Row-wise and Column-wise Sum
public static void rowColumnSum(int[][] matrix) {
    for (int[] row : matrix) {
        System.out.println("Row Sum: " + Arrays.stream(row).sum());
    }
    for (int j = 0; j < matrix[0].length; j++) {
        int colSum = 0;
```

```
            for (int[] ints : matrix) colSum += ints[j];
            System.out.println("Column Sum: " + colSum);
        }
    }
```

```
    // 22. Maximum Element in Matrix
    public static int maxElementMatrix(int[][] matrix) {
        return
Arrays.stream(matrix).flatMapToInt(Arrays::stream).max().orElse(Integer.MIN_
VALUE);
    }
```

```
    // 23. Matrix Multiplication
    public static int[][] multiplyMatrices(int[][] A, int[][] B) {
        int m = A.length, n = B[0].length, p = B.length;
        int[][] result = new int[m][n];
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                for (int k = 0; k < p; k++) {
                    result[i][j] += A[i][k] * B[k][j];
                }
            }
        }
        return result;
    }
```

```
// 24. Rotate Matrix 90 Degrees Clockwise
public static int[][] rotateMatrix(int[][] matrix) {
    int n = matrix.length;
    int[][] rotated = new int[n][n];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            rotated[j][n - i - 1] = matrix[i][j];
        }
    }
    return rotated;
}

// 25. Find Diagonal Sum
public static int diagonalSum(int[][] matrix) {
    int sum = 0, n = matrix.length;
    for (int i = 0; i < n; i++) {
        sum += matrix[i][i] + matrix[i][n - i - 1];
    }
    return sum - (n % 2 == 1 ? matrix[n / 2][n / 2] : 0);
}

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

```
        int[] nums = {2, 3, 5, 7, -7, 5, 8, -5};
        maxProductPair(nums);

        int[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
        printMatrix(matrix);

        int[][] transposed = transposeMatrix(matrix);
        printMatrix(transposed);

        int[][] matrixSum = sumMatrices(matrix, transposed);
        printMatrix(matrixSum);

        rowColumnSum(matrix);

        System.out.println("Max Element: " + maxElementMatrix(matrix));

        int[][] multiplied = multiplyMatrices(matrix, transposed);
        printMatrix(multiplied);

        int[][] rotated = rotateMatrix(matrix);
        printMatrix(rotated);

        System.out.println("Diagonal Sum: " + diagonalSum(matrix));
    }
}
```