Problem No.1

Given an integer array nums of length n, you want to create an array ans of length 2n where

ans[i] == nums[i] and ans[i + n] == nums[i] for 0 <= i < n (0-indexed). Specifically, ans is the concatenation

of two nums arrays. Return the array ans.

Example 1:

Input: nums = [1,2,1]

Output: [1,2,1,1,2,1]

Explanation: The array ans is formed as follows:

- ans = [nums[0],nums[1],nums[2],nums[0],nums[1],nums[2]]

- ans = [1,2,1,1,2,1]

Example 2:

Input: nums = [1,3,2,1]

Output: [1,3,2,1,1,3,2,1]

Explanation: The array ans is formed as follows:

- ans = [nums[0],nums[1],nums[2],nums[3],nums[0],nums[1],nums[2],nums[3]]

- ans = [1,3,2,1,1,3,2,1]

Code:

```
import java.util.Scanner;
import java.util.Arrays;

public class ConcatenateArray {
   public int[] getConcatenation(int[] nums) {
       int n = nums.length;
       int[] ans = new int[2 * n];

       for (int i = 0; i < n; i++) {
           ans[i] = nums[i];
           ans[i + n] = nums[i];
       }

       return ans;
   }
```

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

        System.out.print("Enter the length of the array: ");
        int length = scanner.nextInt();

        int[] nums = new int[length];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < length; i++) {
            nums[i] = scanner.nextInt();
        }

        int[] concatenatedArray = solution.getConcatenation(nums);
        System.out.println("Concatenated Array: " +
Arrays.toString(concatenatedArray));

        scanner.close();
    }
}
```

Problem No.2

You are given an integer array nums containing distinct numbers, and you can perform the following operations until the array is empty:

If the first element has the smallest value, remove it

Otherwise, put the first element at the end of the array.

Return an integer denoting the number of operations it takes to make nums empty.

Example 1:

Input: nums = [3,4,-1]

Output: 5

Operation	Array
1	[4, -1, 3]
2	[-1, 3, 4]
3	[3, 4]
4	[4]
5	[]

Example 2:

Input: nums = [1,2,4,3]

Output: 5

Constraints:

1 <= nums.length <= 105

-109 <= nums[i] <= 109

All values in nums are distinct.

Code:

```
import java.util.*;

public class OperationsToEmptyArray {
   public static int countOperations(int[] nums) {
       int n = nums.length;
       Deque<Integer> deque = new ArrayDeque<>();
       int minIndex = 0;

       for (int i = 1; i < n; i++) {
           if (nums[i] < nums[minIndex]) {
               minIndex = i;
           }
       }

       for (int i = 0; i < n; i++) {
           deque.offerLast(nums[(minIndex + i) % n]);
```

```java
        }

        int operations = 0;
        int expected = nums[minIndex];

        // Perform operations until the deque is empty
        while (!deque.isEmpty()) {
            if (deque.peekFirst() == expected) {
                deque.pollFirst();
                operations++;
                if (!deque.isEmpty()) {
                    expected = Math.min(expected, deque.peekFirst());
                }
            } else {
                deque.offerLast(deque.pollFirst());
            }
        }

        return operations;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of elements: ");
        int n = scanner.nextInt();
        int[] nums = new int[n];
        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }
        int result = countOperations(nums);
        System.out.println("Number of operations to make nums empty: " +
result);
        scanner.close();
    }
}
```

Problem No.3

You are given a 0-indexed 1-dimensional (1D) integer array original, and two integers, m and n. You are tasked with creating a 2-dimensional (2D) array with m rows and n columns using all the elements from original.

The elements from indices 0 to n - 1 (inclusive) of original should form the first row of the constructed 2D array, the elements from indices n to 2 * n - 1 (inclusive) should form the second row of the constructed 2D array, and so on.

Return an m x n 2D array constructed according to the above procedure, or an empty 2D array if it is impossible.

Example 1:

Input: original = [1,2,3,4], m = 2, n = 2

Output: [[1,2],[3,4]]

Explanation: The constructed 2D array should contain 2 rows and 2 columns.

The first group of n=2 elements in original, [1,2], becomes the first row in the constructed 2D array.

The second group of n=2 elements in original, [3,4], becomes the second row in the constructed 2D array.

Example 2:

Input: original = [1,2,3], m = 1, n = 3

Output: [[1,2,3]]

Explanation: The constructed 2D array should contain 1 row and 3 columns.

Put all three elements in original into the first row of the constructed 2D array.

Example 3:

Input: original = [1,2], m = 1, n = 1

Output: []

Explanation: There are 2 elements in original.

It is impossible to fit 2 elements in a 1x1 2D array, so return an empty 2D array.

Constraints:

1 <= original.length <= 5 * 104

1 <= original[i] <= 105

1 <= m, n <= 4 * 104

Code:

```
import java.util.*;

public class Construct2DArray {
   public static void main(String[] args) {
       Scanner scanner = new Scanner(System.in);

       System.out.println("Enter the elements of the original array separated
by spaces:");
       String[] input = scanner.nextLine().split(" ");
```

```java
        int[] original = new int[input.length];
        for (int i = 0; i < input.length; i++) {
            original[i] = Integer.parseInt(input[i]);
        }

        System.out.println("Enter the number of rows (m):");
        int m = scanner.nextInt();

        System.out.println("Enter the number of columns (n):");
        int n = scanner.nextInt();

        int[][] result = construct2DArray(original, m, n);

        if (result.length == 0) {
            System.out.println("Empty 2D array");
        } else {
            System.out.println("Constructed 2D array:");
            for (int[] row : result) {
                System.out.println(Arrays.toString(row));
            }
        }

        scanner.close();
    }

    public static int[][] construct2DArray(int[] original, int m, int n) {
        if (original.length != m * n) {
            return new int[0][0]; // Empty 2D array
        }

        int[][] result = new int[m][n];
        int index = 0;

        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                result[i][j] = original[index++];
            }
        }

        return result;
    }
}
```

Problem No.4

You are given an integer array nums. You should move each element of nums into one of the two arrays A and B such that A and B are non-empty, and average(A) == average(B).

Return true if it is possible to achieve that and false otherwise.

Note that for an array arr, average(arr) is the sum of all the elements of arr over the length of arr.

Example 1:

Input: nums = [1,2,3,4,5,6,7,8]

Output: true

Explanation: We can split the array into [1,4,5,8] and [2,3,6,7], and both of them have an average of 4.5.

Example 2:

Input: nums = [3,1]

Output: false

Code:

```
import java.util.*;

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

       System.out.println("Enter the elements of the array nums separated by
spaces:");
       String[] input = scanner.nextLine().split(" ");
       int[] nums = new int[input.length];
       for (int i = 0; i < input.length; i++) {
           nums[i] = Integer.parseInt(input[i]);
       }

       boolean result = canSplitArray(nums);

       System.out.println("Output: " + result);

       scanner.close();
   }

   public static boolean canSplitArray(int[] nums) {
       int sum = 0;
       for (int num : nums) {
           sum += num;
```

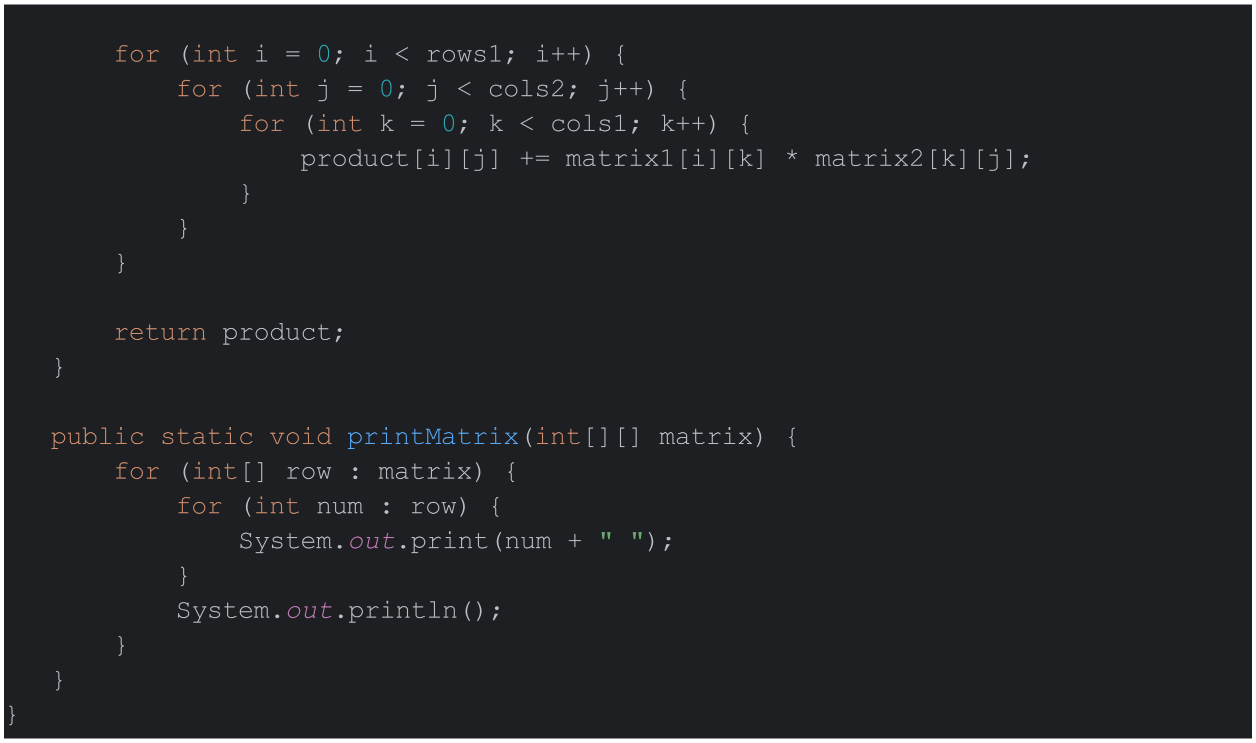
```
        }

        Arrays.sort(nums);

        int n = nums.length;
        for (int lenA = 1; lenA <= n / 2; lenA++) {
            if (sum * lenA % n == 0 && subsetSum(nums, sum * lenA / n, lenA, 0))
{
                return true;
            }
        }

        return false;
    }

    private static boolean subsetSum(int[] nums, int targetSum, int k, int
startIndex) {
        if (k == 0) {
            return targetSum == 0;
        }

        for (int i = startIndex; i < nums.length - k + 1; i++) {
            if (i > startIndex && nums[i] == nums[i - 1]) {
                continue;
            }
            if (nums[i] <= targetSum && subsetSum(nums, targetSum - nums[i], k -
1, i + 1)) {
                return true;
            }
        }

        return false;
    }
}
```

Problem No.5

Write a program that takes two sizes of two different matrices. Check if matrix multiplication is possible or not for the given sizes. If matrix multiplication is possible then return the product matrix as result.

Code:

```
import java.util.Scanner;

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

        System.out.println("Enter the number of rows for the first matrix:");
        int rows1 = scanner.nextInt();

        System.out.println("Enter the number of columns for the first matrix:");
        int cols1 = scanner.nextInt();

        System.out.println("Enter the number of rows for the second matrix:");
        int rows2 = scanner.nextInt();

        System.out.println("Enter the number of columns for the second
matrix:");
        int cols2 = scanner.nextInt();

        int[][] matrix1 = new int[rows1][cols1];
        int[][] matrix2 = new int[rows2][cols2];

        if (cols1 != rows2) {
            System.out.println("Matrix multiplication is not possible with the
given sizes.");
        } else {
            System.out.println("Enter the elements of the first matrix:");
            inputMatrix(scanner, matrix1);

            System.out.println("Enter the elements of the second matrix:");
            inputMatrix(scanner, matrix2);

            int[][] product = multiplyMatrices(matrix1, matrix2);

            System.out.println("Product of the matrices:");
            printMatrix(product);
        }

        scanner.close();
    }

    public static void inputMatrix(Scanner scanner, int[][] matrix) {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[0].length; j++) {
                matrix[i][j] = scanner.nextInt();
            }
        }
    }

    public static int[][] multiplyMatrices(int[][] matrix1, int[][] matrix2) {
        int rows1 = matrix1.length;
        int cols1 = matrix1[0].length;
        int cols2 = matrix2[0].length;

        int[][] product = new int[rows1][cols2];
```



```java
        for (int i = 0; i < rows1; i++) {
            for (int j = 0; j < cols2; j++) {
                for (int k = 0; k < cols1; k++) {
                    product[i][j] += matrix1[i][k] * matrix2[k][j];
                }
            }
        }

        return product;
    }

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