Program 1

```
import java.util.Scanner;
public class ConcatenateArrays {
    public static int[] concatenateArrays(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) {
        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();
        }
        scanner.close();
        int[] concatenatedArray = concatenateArrays(nums);
        System.out.print("Concatenated Array: ");
        for (int num : concatenatedArray) {
            System.out.print(num + " ");
        }
    }
}
```

Program2

```
import java.util.Arrays;
import java.util.PriorityQueue;
public class EmptyArrayOperations {
    public static void main(String[] args) {
        int[] nums = {3, 4, -1};
        System.out.println(minOperations(nums));}
    public static int minOperations(int[] nums) {
        PriorityQueue<Integer> minHeap = new PriorityQueue<>();
        for (int num : nums) {
            minHeap.offer(num);
        }
        int operations = 0;
        int n = nums.length;
        int index = 0;
        while (index < n) {
            if (nums[index] == minHeap.peek()) {
                minHeap.poll();
                index++;
                operations++;
            } else {
                int firstElement = nums[index];
                for (int i = index; i < n - 1; i++) {
                    nums[i] = nums[i + 1];
                }
                nums[n - 1] = firstElement;
                operations++;
            } }
        return operations;
 }}
```

Program 3

```
import java.util.Scanner;


public class Construct2DArray {
    public static int[][] construct2DArray(int[] original, int m, int n) {
        int totalElements = m * n;
        if (original.length != totalElements) {
            System.out.println("It's impossible to construct a " + m + " x " + n + " 2D array.");
            return new int[0][0];
        }
        int[][] result = new int[m][n];
        for (int i = 0; i < totalElements; i++) {
            result[i / n][i % n] = original[i];
        }
        return result;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the length of the original array: ");
        int length = scanner.nextInt();
        int[] original = new int[length];
        System.out.println("Enter the elements of the original array:");
        for (int i = 0; i < length; i++) {
            original[i] = scanner.nextInt();
        }
        System.out.print("Enter the number of rows (m): ");
        int m = scanner.nextInt();
```

```
        System.out.print("Enter the number of columns (n): ");
        int n = scanner.nextInt();
        scanner.close();
        int[][] result = construct2DArray(original, m, n);
        System.out.println("Constructed 2D Array:");
        for (int[] row : result) {
            for (int element : row) {
                System.out.print(element + " ");
            }
            System.out.println();
        }
    }
}
```

Program 4

```
import java.util.Arrays;
public class SplitArrayWithEqualAverage {
   public static void main(String[] args) {
      int[] nums = {3,1};
      System.out.println(canSplitArray(nums));
   }
   public static boolean canSplitArray(int[] nums) {
      int sum = Arrays.stream(nums).sum();
      int n = nums.length;
      for (int i = 1; i < n; i++) {
    if (sum * i % n == 0 && canSplit(nums, 0, sum * i / n, i))
    {
            return true;
          }
      }
      return false;
   }
   private static boolean canSplit(int[] nums, int index, int targetSum, int k) {
      if (k == 0) {
         return targetSum == 0;
      }
      if (index == nums.length) {
         return false;
      }
      if (nums[index] <= targetSum &&
            canSplit(nums, index + 1, targetSum - nums[index], k - 1)) {
         return true;
      }
```

```
        if (canSplit(nums, index + 1, targetSum, k)) {
            return true;
        }

        return false;
    }
}
```

Program 5

```
import java.util.Scanner;
public class MatrixMultiplication {
  public static void main(String[] args) {
   Scanner scanner = new Scanner(System.in);
       System.out.println("Enter the size of the first matrix (rows and columns):");
       int rows1 = scanner.nextInt();
       int cols1 = scanner.nextInt();
       int[][] matrix1 = inputMatrix(rows1, cols1, "first");
       System.out.println("Enter the size of the second matrix (rows and columns):");
       int rows2 = scanner.nextInt();
       int cols2 = scanner.nextInt();
       int[][] matrix2 = inputMatrix(rows2, cols2, "second");
       if (cols1 == rows2) {
          int[][] resultMatrix = multiplyMatrices(matrix1, matrix2);
          System.out.println("Matrix multiplication is possible, and the result is:");
          printMatrix(resultMatrix);
       }
else
{
          System.out.println("Matrix multiplication is not possible for the given sizes.");
       }
    scanner.close();
   }
   private static int[][] inputMatrix(int rows, int cols, String matrixName) {
       Scanner scanner = new Scanner(System.in);
       System.out.println("Enter the elements for the " + matrixName + " matrix:");
       int[][] matrix = new int[rows][cols];
       for (int i = 0; i < rows; i++) {
```

```
        for (int j = 0; j < cols; j++) {
            matrix[i][j] = scanner.nextInt();
        }}
        return matrix;
    }
    private static int[][] multiplyMatrices(int[][] matrix1, int[][] matrix2) {
        int rows1 = matrix1.length;
        int cols1 = matrix1[0].length;
        int cols2 = matrix2[0].length;
    int[][] resultMatrix = new int[rows1][cols2];
   for (int i = 0; i < rows1; i++) {
            for (int j = 0; j < cols2; j++) {
                for (int k = 0; k < cols1; k++) {
                    resultMatrix[i][j] += matrix1[i][k] * matrix2[k][j];
                }}}
   return resultMatrix;
    }
    private static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int element : row) {
                System.out.print(element + " "); }
            System.out.println();
   }
    }
}
```