Answer 1)

```
/* Online Java Compiler and Editor */
import java.util.ArrayList;
import java.util.List;

public class HelloWorld {
    public static List<Integer> findCommonElements(int[] arr1, int[] arr2, int[] arr3) {
        List<Integer> result = new ArrayList<>();

        int i = 0, j = 0, k = 0;
        while (i < arr1.length && j < arr2.length && k < arr3.length) {
            if (arr1[i] == arr2[j] && arr2[j] == arr3[k]) {
                result.add(arr1[i]);
                i++;
                j++;
                k++;
            } else if (arr1[i] < arr2[j]) {
                i++;
            } else if (arr2[j] < arr3[k]) {
                j++;
            } else {
                k++;
            }
        }

        return result;
    }

    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3, 4, 5};
        int[] arr2 = {1, 2, 5, 7, 9};
        int[] arr3 = {1, 3, 4, 5, 8};

        List<Integer> commonElements = findCommonElements(arr1, arr2, arr3);
        System.out.println(commonElements);
    }
}
```

Answer 2)

```
/* Online Java Compiler and Editor */
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class HelloWorld {
    public static List<List<Integer>> findDistinctIntegers(int[] nums1, int[] nums2) {
        List<List<Integer>> result = new ArrayList<>();
        Set<Integer> set1 = new HashSet<>();
        Set<Integer> set2 = new HashSet<>();

        for (int num : nums1) {
            set1.add(num);
        }

        for (int num : nums2) {
            set2.add(num);
        }

        List<Integer> distinctNums1 = new ArrayList<>();
        List<Integer> distinctNums2 = new ArrayList<>();

        for (int num : set1) {
            if (!set2.contains(num)) {
                distinctNums1.add(num);
            }
        }

        for (int num : set2) {
            if (!set1.contains(num)) {
                distinctNums2.add(num);
            }
        }

        result.add(distinctNums1);
        result.add(distinctNums2);

        return result;
    }

    public static void main(String[] args) {
```

```
        int[] nums1 = {1, 2, 3};
        int[] nums2 = {2, 4, 6};

        List<List<Integer>> distinctIntegers = findDistinctIntegers(nums1, nums2);
        System.out.println(distinctIntegers);
    }
}
```

Answer 3)

```
/* Online Java Compiler and Editor */
public class MatrixTranspose {
    public static int[][] transpose(int[][] matrix) {
        int rows = matrix.length;
        int cols = matrix[0].length;

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

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                transposeMatrix[j][i] = matrix[i][j];
            }
        }

        return transposeMatrix;
    }

    public static void main(String[] args) {
        int[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

        int[][] transposedMatrix = transpose(matrix);

        // Print the transposed matrix
        for (int[] row : transposedMatrix) {
            for (int num : row) {
                System.out.print(num + " ");
            }
            System.out.println();
        }
    }
}
```

Answer 4)

```
/* Online Java Compiler and Editor */
import java.util.Arrays;

public class HelloWorld {
    public static int arrayPairSum(int[] nums) {
        Arrays.sort(nums);  // Sort the array in ascending order

        int sum = 0;
        for (int i = 0; i < nums.length; i += 2) {
            sum += nums[i];
        }

        return sum;
    }

    public static void main(String[] args) {
        int[] nums = {1, 4, 3, 2};
        int maxSum = arrayPairSum(nums);
        System.out.println(maxSum);
    }
}
```

Answer 5)

```
/* Online Java Compiler and Editor */
public class HelloWorld {
    public static int arrangeCoins(int n) {
        long left = 0;
        long right = n;

        while (left <= right) {
            long mid = left + (right - left) / 2;
            long curr = mid * (mid + 1) / 2;

            if (curr == n) {
                return (int) mid;
            } else if (curr < n) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
```

```
        }

        return (int) right;
    }

    public static void main(String[] args) {
        int n = 8;
        int completeRows = arrangeCoins(n);
        System.out.println(completeRows);
    }
}
```

Answer 6)

```
/* Online Java Compiler and Editor */
import java.util.Arrays;

public class HelloWorld {
    public static int[] sortedSquares(int[] nums) {
        int n = nums.length;
        int[] result = new int[n];

        int left = 0;
        int right = n - 1;

        for (int i = n - 1; i >= 0; i--) {
            if (Math.abs(nums[left]) > Math.abs(nums[right])) {
                result[i] = nums[left] * nums[left];
                left++;
            } else {
                result[i] = nums[right] * nums[right];
                right--;
            }
        }

        return result;
    }

    public static void main(String[] args) {
        int[] nums = {-4, -1, 0, 3, 10};
        int[] squaredArray = sortedSquares(nums);
        System.out.println(Arrays.toString(squaredArray));
    }
```

```
}
```

Answer 7)

```
/* Online Java Compiler and Editor */
public class HelloWorld {
    public static int maxCount(int m, int n, int[][] ops) {
        int minRow = m;
        int minCol = n;

        for (int[] op : ops) {
            minRow = Math.min(minRow, op[0]);
            minCol = Math.min(minCol, op[1]);
        }

        return minRow * minCol;
    }

    public static void main(String[] args) {
        int m = 3;
        int n = 3;
        int[][] ops = {{2, 2}, {3, 3}};

        int maxIntegers = maxCount(m, n, ops);
        System.out.println(maxIntegers);
    }
}
```

Answer 8)

```
/* Online Java Compiler and Editor */
import java.util.Arrays;

public class HelloWorld {
    public static int[] shuffle(int[] nums, int n) {
        int[] result = new int[2 * n];

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

        return result;
```

```
    }

    public static void main(String[] args) {
        int[] nums = {2, 5, 1, 3, 4, 7};
        int n = 3;

        int[] shuffledArray = shuffle(nums, n);
        System.out.println(Arrays.toString(shuffledArray));
    }
}
```