Task 2

a)

import java.util.\*;

public class IntervalIntersection {

public static int[][] intervalIntersection(int[][] A, int[][] B) {

List<int[]> result = new ArrayList<>();

int i = 0, j = 0;

while (i < A.length && j < B.length) {

// Find overlap

int start = Math.max(A[i][0], B[j][0]);

int end = Math.min(A[i][1], B[j][1]);

if (start <= end) {

result.add(new int[]{start, end});

}

// Move pointer with smaller endpoint

if (A[i][1] < B[j][1]) {

i++;

} else {

j++;

}

}

return result.toArray(new int[result.size()][]);

}

public static void main(String[] args) {

int[][] A = {{0, 2}, {5, 10}, {13, 23}, {24, 25}};

int[][] B = {{1, 5}, {8, 12}, {15, 24}, {25, 26}};

int[][] result = intervalIntersection(A, B);

for (int[] r : result) {

System.out.println("[" + r[0] + ", " + r[1] + "]");

}

}

}

b)

import java.util.\*;

public class MergeSortedArrays {

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

int n1 = arr1.length, n2 = arr2.length;

int[] result = new int[n1 + n2];

int i = 0, j = 0, k = 0;

while (i < n1 && j < n2) {

if (arr1[i] <= arr2[j]) {

result[k++] = arr1[i++];

} else {

result[k++] = arr2[j++];

}

}

while (i < n1) result[k++] = arr1[i++];

while (j < n2) result[k++] = arr2[j++];

return result;

}

public static void main(String[] args) {

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

int[] arr2 = {2, 4, 6, 8};

int[] merged = merge(arr1, arr2);

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

}

}

c)

import java.util.\*;

public class ThreeSum {

public static List<List<Integer>> threeSum(int[] nums) {

Arrays.sort(nums);

List<List<Integer>> result = new ArrayList<>();

for (int i = 0; i < nums.length - 2; i++) {

if (i > 0 && nums[i] == nums[i - 1]) continue; // skip duplicates

int left = i + 1, right = nums.length - 1;

while (left < right) {

int sum = nums[i] + nums[left] + nums[right];

if (sum == 0) {

result.add(Arrays.asList(nums[i], nums[left], nums[right]));

// Skip duplicates

while (left < right && nums[left] == nums[left + 1]) left++;

while (left < right && nums[right] == nums[right - 1]) right--;

left++;

right--;

} else if (sum < 0) {

left++;

} else {

right--;

}

}

}

return result;

}

public static void main(String[] args) {

int[] nums = {-1, 0, 1, 2, -1, -4};

List<List<Integer>> triplets = threeSum(nums);

System.out.println("Unique triplets with sum = 0: " + triplets);

}

}