# Q1. <https://leetcode.com/problems/move-zeroes/>

## Solution:

import java.util.Arrays;  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 29/05/2023  
 \*/*public class QuestionOneSolution {  
 public static void main(String[] args) {  
 int[] nums1 = {0,1,0,3,12};  
 *moveZeros*(nums1);  
 System.*out*.println(Arrays.*toString*(nums1));  
 }  
  
 static void moveZeros(int[] nums) {  
 int count = 0;  
 for (int i = 0; i < nums.length; i++) {  
 if (nums[i] != 0) {  
 nums[count++] = nums[i];  
 }  
 }  
 while (count < nums.length) {  
 nums[count++] = 0;  
 }  
 }  
}

# Q2. <https://leetcode.com/problems/contains-duplicate/>

## Solution*:*

import java.util.HashSet;  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 29/05/2023  
 \*/*public class QuestionTwoSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*containsDuplicate*(new int[]{1, 2, 3, 1}));  
 System.*out*.println(*containsDuplicate*(new int[]{1, 2, 3, 4}));  
 System.*out*.println(*containsDuplicate*(new int[]{1, 1, 1, 3, 3, 4, 3, 2, 4, 2}));  
 }  
  
 static boolean containsDuplicate(int[] nums) {  
 HashSet<Integer> set = new HashSet<>();  
 for (int num : nums) {  
 if (set.contains(num)) {  
 return true;  
 }  
 set.add(num);  
 }  
 return false;  
 }  
}

# Q3. <https://leetcode.com/problems/contains-duplicate-ii/>

## Solution:

import java.util.HashMap;  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 29/05/2023  
 \*/*public class QuestionThreeSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*containsNearbyDuplicate*(new int[]{1, 2, 3, 1}, 3));  
 System.*out*.println(*containsNearbyDuplicate*(new int[]{1, 0, 1, 1}, 1));  
 System.*out*.println(*containsNearbyDuplicate*(new int[]{1, 2, 3, 1, 2, 3}, 2));  
 }  
  
 static boolean containsNearbyDuplicate(int[] nums, int k) {  
 if (k == 0) return false;  
 HashMap<Integer, Integer> map = new HashMap<>();  
 for (int i = 0; i < nums.length; i++) {  
 if ((map.containsKey(nums[i])) && (i - map.get(nums[i]) <= k)) {  
 return true;  
 }  
 map.put(nums[i], i);  
 }  
 return false;  
 }  
}

# Q4. <https://leetcode.com/problems/summary-ranges/>

## Solution:

import java.util.ArrayList;  
import java.util.List;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 29/05/2023  
 \*/*public class QuestionFourSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*summaryRanges*(new int[]{0, 1, 2, 4, 5, 7}));  
 System.*out*.println(*summaryRanges*(new int[]{0, 2, 3, 4, 6, 8, 9}));  
 }  
  
 static List<String> summaryRanges(int[] nums) {  
 List<String> list = new ArrayList<>();  
 for (int i = 0; i < nums.length; i++) {  
 int start = nums[i];  
 while (i + 1 < nums.length && nums[i] + 1 == nums[i + 1]) i++;  
 if (start != nums[i]) {  
 list.add(start + "->" + nums[i]);  
 } else {  
 list.add("" + start);  
 }  
 }  
 return list;  
 }  
}

# Q5. <https://leetcode.com/problems/pascals-triangle-ii/>

## Solution:

# Q6. <https://leetcode.com/problems/best-time-to-buy-and-sell-stock/>

## Solution:

# Q7. <https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/>

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 29/05/2023  
 \*/*public class QuestionSevenSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*removeElement*(new int[]{3, 2, 2, 3}, 3));  
 System.*out*.println(*removeElement*(new int[]{0, 1, 2, 2, 3, 0, 4, 2}, 2));  
 }  
  
 static int removeElement(int[] nums, int val) {  
 int k = 0;  
 for (int i = 0; i < nums.length; i++) {  
 if (nums[i] != val) {  
 int temp = nums[k];  
 nums[k] = nums[i];  
 nums[i] = temp;  
 k++;  
 }  
 }  
 System.*out*.println(Arrays.*toString*(nums));  
 return k;  
 }  
}

# Q8. <https://leetcode.com/problems/majority-element/>

## Solution:

import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 29/05/2023  
 \*/*public class QuestionEightSolution {  
 public static void main(String[] args) {  
 System.*out*.println(Arrays.*toString*(*intersection*(new int[]{1, 2, 2, 1}, new int[]{2, 2})));  
 System.*out*.println(Arrays.*toString*(*intersection*(new int[]{4, 9, 5}, new int[]{9, 4, 9, 8, 4})));  
 }  
  
 static int[] intersection(int[] nums1, int[] nums2) {  
 List<Integer> list = new ArrayList<>();  
 int[] count = new int[20];  
 for (int num : nums1) {  
 count[num]++;  
 }  
 for (int num : nums2) {  
 if (count[num] != 0) {  
 list.add(num);  
 count[num] = 0;  
 }  
 }  
 int[] result = new int[list.size()];  
 for (int i = 0; i < result.length; i++) {  
 result[i] = list.get(i);  
 }  
 return result;  
 }  
}

# Q9. <https://leetcode.com/problems/majority-element-ii/>

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 29/05/2023  
 \*/*public class QuestionNineSolution {  
 public static void main(String[] args) {  
 System.*out*.println(Arrays.*toString*(*intersect*(new int[]{1, 2, 2, 1}, new int[]{2, 2})));  
 System.*out*.println(Arrays.*toString*(*intersect*(new int[]{4, 9, 5}, new int[]{9, 4, 9, 8, 4})));  
 }  
  
 static int[] intersect(int[] nums1, int[] nums2) {  
 Arrays.*sort*(nums1);  
 Arrays.*sort*(nums2);  
 int i = 0, j = 0, k = 0;  
 while (i < nums1.length && j < nums2.length) {  
 if (nums1[i] < nums2[j]) {  
 i++;  
 } else if (nums1[i] > nums2[j]) {  
 j++;  
 } else {  
 nums1[k++] = nums1[i++];  
 j++;  
 }  
 }  
 return Arrays.*copyOfRange*(nums1, 0, k);  
 }  
}

# Q10. <https://leetcode.com/problems/missing-ranges/>

## Solution:

import java.util.ArrayList;  
import java.util.HashSet;  
import java.util.List;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 23/05/2023  
 \*/*public class QuestionTenSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*findMissingElements*(new int[]{3, 1, 2, 4, 10}, 2, 10));  
 }  
  
 static List<Integer> findMissingElements(int[] nums, int low, int high) {  
 HashSet<Integer> set = new HashSet<>();  
 List<Integer> list = new ArrayList<>();  
 for (int num : nums) {  
 set.add(num);  
 }  
 for (int i = low; i <= high; i++) {  
 if (!set.contains(i)) {  
 list.add(i);  
 }  
 }  
 return list;  
 }  
}

# Q11. <https://leetcode.com/problems/3sum/>

## Solution:

import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.HashSet;  
import java.util.List;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 24/05/2023  
 \*/*public class QuestionElevenSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*findThreeSum*(new int[]{-1, 0, 1, 2, -1, -4}));  
 }  
  
 static List<List<Integer>> findThreeSum(int[] nums) {  
 HashSet<List<Integer>> result = new HashSet<>();  
 if (nums.length == 0) return new ArrayList<>(result);  
 Arrays.*sort*(nums);  
 for (int i = 0; i < nums.length - 2; i++) {  
 int j = i + 1;  
 int k = nums.length - 1;  
 while (j < k) {  
 int sum = nums[j] + nums[k];  
 if (sum == -nums[i]) {  
 result.add(Arrays.*asList*(nums[i], nums[j], nums[k]));  
 j++;  
 k--;  
 } else if (sum > -nums[i]) k--;  
 else if (sum < -nums[i]) j++;  
 }  
 }  
 return new ArrayList<>(result);  
 }  
}

# Q13. <https://leetcode.com/problems/3sum-closest/>

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 24/05/2023  
 \*/*public class QuestionThirteenSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*threeSumClosest*(new int[]{-1, 2, 1, -4},1));  
 }  
  
 static int threeSumClosest(int[] nums, int target) {  
 int min = Integer.*MAX\_VALUE*;  
 int result = 0;  
 Arrays.*sort*(nums);  
 for (int i = 0; i < nums.length - 2; i++) {  
 int j = i + 1;  
 int k = nums.length - 1;  
 while (j < k) {  
 int sum = nums[i] + nums[j] + nums[k];  
 int diff = Math.*abs*(sum - target);  
 if (diff == 0) return sum;  
 if (diff < min) {  
 min = diff;  
 result = sum;  
 }  
 if (sum < target) j++;  
 else k--;  
 }  
 }  
 return result;  
 }  
}

# Q14. <https://leetcode.com/problems/4sum/>

## Solution:

import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.HashSet;  
import java.util.List;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 27/05/2023  
 \*/*public class QuestionFourteenSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*findFourSum*(new int[]{1, 0, -1, 0, -2, 2}, 0));  
 System.*out*.println(*findFourSum*(new int[]{-1, 0, 1, 2, -1, -4}, -1));  
 }  
  
 static List<List<Integer>> findFourSum(int[] nums, int target) {  
 List<List<Integer>> result = new ArrayList<>();  
 int n = nums.length;  
 Arrays.*sort*(nums);  
 for (int i = 0; i < n - 2; i++) {  
 if (i > 0 && nums[i] == nums[i - 1]) {  
 continue;  
 }  
 for (int j = i + 1; j < n - 2; j++) {  
 if (j > i + 1 && nums[j] == nums[j - 1]) {  
 continue;  
 }  
 int left = j + 1;  
 int right = n - 1;  
 while (left < right) {  
 long sum = (long) nums[i] + nums[j] + nums[left] + nums[right];  
 if (sum == target) {  
 result.add(Arrays.*asList*(nums[i], nums[j], nums[left], nums[right]));  
 while (left < right && nums[left] == nums[left + 1]) left++;  
 while (left < right && nums[right] == nums[right - 1]) right--;  
 left++;  
 right--;  
 } else if (sum < target) {  
 left++;  
 } else if (sum > target) {  
 right--;  
 }  
 }  
 }  
 }  
 return new ArrayList<>(result);  
 }  
}

# Q15. <https://leetcode.com/problems/rotate-image/>

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 23/05/2023  
 \*/*public class QuestionFifteenSolution {  
 public static void main(String[] args) {  
 *rotate*(new int[][]{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}});  
 }  
  
 static void rotate(int[][] matrix) {  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 System.*out*.print(matrix[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 System.*out*.println();  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = i; j < matrix.length; j++) {  
 int temp = matrix[i][j];  
 matrix[i][j] = matrix[j][i];  
 matrix[j][i] = temp;  
 }  
 }  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix.length/2; j++) {  
 int temp = matrix[i][j];  
 matrix[i][j] = matrix[i][matrix.length - 1 - j];  
 matrix[i][matrix.length - 1 - j] = temp;  
 }  
 }  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 System.*out*.print(matrix[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 }  
}