# 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/range-sum-query-immutable/>

## Solution:

public class NumArray {  
 int[] num;  
  
 public NumArray(int[] nums) {  
 num = new int[nums.length];  
 for (int i = 0; i < nums.length; i++) {  
 num[i] = nums[i];  
 }  
 }  
  
 public int sumRange(int left, int right) {  
 int sum = 0;  
 for (int i = left; i <= right; i++) {  
 sum += num[i];  
 }  
 return sum;  
 }  
}

# Q6. <https://leetcode.com/problems/range-sum-query-2d-immutable/>

## Solution:

# Q7. <https://leetcode.com/problems/remove-element/>

## 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/intersection-of-two-arrays/>

## 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/intersection-of-two-arrays-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/next-greater-element-i/>

## Solution:

import java.util.Arrays;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Stack;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 30/05/2023  
 \*/*public class QuestionTenSolution {  
 public static void main(String[] args) {  
 System.*out*.println(Arrays.*toString*(*nextGreaterElement*(new int[]{4, 1, 2}, new int[]{1, 3, 4, 2})));  
 System.*out*.println(Arrays.*toString*(*nextGreaterElement*(new int[]{4, 1, 2}, new int[]{1, 2, 3, 4})));  
 System.*out*.println(Arrays.*toString*(*nextGreaterElement*(new int[]{2, 4}, new int[]{1, 2, 3, 4})));  
 }  
  
 static int[] nextGreaterElement(int[] nums1, int[] nums2) {  
 Map<Integer, Integer> map = new HashMap<>();  
 Stack<Integer> stack = new Stack<>();  
 for (int num : nums2) {  
 while(!stack.empty() && stack.peek()<num){  
 map.put(stack.pop(), num);  
 }  
 stack.push(num);  
 }  
 System.*out*.println(map);  
 for (int i = 0; i < nums1.length; i++) {  
 nums1[i] = map.getOrDefault(nums1[i], -1);  
 }  
 return nums1;  
 }  
}

# Q11. <https://leetcode.com/problems/next-greater-element-ii/>

## Solution:

import java.util.Arrays;  
import java.util.Stack;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 31/05/2023  
 \*/*public class QuestionElevenSolution {  
 public static void main(String[] args) {  
 System.*out*.println(Arrays.*toString*(*nextGreaterElements*(new int[]{1, 2, 3, 4, 3})));  
 System.*out*.println(Arrays.*toString*(*nextGreaterElements*(new int[]{5,4,3,2,1})));  
 }  
  
 static int[] nextGreaterElements(int[] nums) {  
 Stack<Integer> stack = new Stack<>();  
 int n = nums.length;  
 int[] result = new int[n];  
 for (int i = (2 \* n)-1; i >= 0; i--) {  
 while (!stack.empty() && stack.peek() <= nums[i % n]) {  
 stack.pop();  
 }  
 if (i < n) {  
 if (!stack.empty()) {  
 result[i] = stack.peek();  
 } else {  
 result[i] = -1;  
 }  
 }  
 stack.push(nums[i % n]);  
 }  
 return result;  
 }  
}

# Q12. <https://leetcode.com/problems/next-greater-element-iii/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 31/05/2023  
 \* https://leetcode.com/problems/next-greater-element-iii/solutions/3580067/java-best-solution-0-ms-beats-100-linear-time-comeplexity/  
 \*/*public class QuestionTwelveSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*nextGreaterElementThree*(12));  
 System.*out*.println(*nextGreaterElementThree*(125431));  
 }  
  
 static int nextGreaterElementThree(int n) {  
 char[] str = (n + "").toCharArray();  
 int deflectionPoint = str.length - 1;  
 *//find out the deflection point* while (deflectionPoint > 0) {  
 if (str[deflectionPoint] > str[deflectionPoint - 1]) {  
 break;  
 }  
 deflectionPoint--;  
 }  
  
 *//if deflection point is 0 return -1* if (deflectionPoint == 0) return -1;  
  
 *//find out 1st and 2nd swapping index* int firstSwappingIndex = deflectionPoint - 1;  
 int secondSwappingIndex = str.length - 1;  
  
 while (firstSwappingIndex <= secondSwappingIndex) {  
 if (str[firstSwappingIndex] < str[secondSwappingIndex]) {  
 break;  
 }  
 secondSwappingIndex--;  
 }  
  
 *//swap two elements* char temp = str[firstSwappingIndex];  
 str[firstSwappingIndex] = str[secondSwappingIndex];  
 str[secondSwappingIndex] = temp;  
  
 *//reverse the rest element at point of deflection  
 reverseChar*(str, deflectionPoint);  
  
 *//convert the string to long and check Integer.MAX\_VALUE and then return* long result = Long.*parseLong*(new String(str));  
 if (result > Integer.*MAX\_VALUE*) return -1;  
 return (int) result;  
 }  
  
 static void reverseChar(char[] str, int i) {  
 int start = i;  
 int end = str.length - 1;  
 while (start <= end) {  
 char temp = str[start];  
 str[start] = str[end];  
 str[end] = temp;  
 start++;  
 end--;  
 }  
 }  
}

# Q13. <https://leetcode.com/problems/4sum/>

## Solution:

# Q14. <https://leetcode.com/problems/reduce-array-size-to-the-half/>

## Solution:

import java.util.\*;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 02/06/2023  
 \*/*public class QuestionThirteenSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*minSetSize*(new int[]{3, 3, 3, 3, 5, 5, 5, 2, 2, 7}));  
 }  
  
 static int minSetSize(int[] arr) {  
 Map<Integer, Integer> map = new HashMap<>();  
 for (int num : arr) {  
 map.put(num, map.getOrDefault(num, 0) + 1);  
 }  
 List<Integer> freqs = new ArrayList<>(map.values());  
 freqs.sort(Collections.*reverseOrder*());  
 int count = 0;  
 int totalElement = arr.length;  
 int i = 0;  
 while (totalElement > arr.length / 2) {  
 totalElement = totalElement - freqs.get(i);  
 i++;  
 count++;  
 }  
 return count;  
 }  
}