Q1. <https://leetcode.com/problems/find-common-characters/>

## Solution:

import java.util.ArrayList;  
import java.util.List;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 05/06/2023  
 \*/*public class FindCommonCharacters {  
 public static void main(String[] args) {  
 System.*out*.println(*commonStrings*(new String[]{"bella", "label", "roller"}));  
 }  
  
 public static List<String> commonStrings(String[] words) {  
 List<String> res = new ArrayList<>();  
 String temp = words[0];  
 for (int i = 1; i < words.length; i++) {  
 temp = *getCommonString*(temp, words[i]);  
 if (temp.length() == 0) return res;  
 }  
 char[] arr = temp.toCharArray();  
 for (char c : arr) {  
 res.add(String.*valueOf*(c));  
 }  
 return res;  
 }  
  
 public static String getCommonString(String a, String b) {  
 int[] arr1 = new int[26];  
 int[] arr2 = new int[26];  
 for (char c : a.toCharArray()) {  
 arr1[c - 'a']++;  
 }  
 for (char c : b.toCharArray()) {  
 arr2[c - 'a']++;  
 }  
 StringBuilder sb = new StringBuilder();  
 for (int i = 0; i < 26; i++) {  
 if (arr1[i] != 0 && arr2[i] != 0) {  
 for (int j = 0; j < Math.*min*(arr1[i], arr2[i]); j++) {  
 sb.append((char) (i + 'a'));  
 }  
 }  
 }  
 return sb.toString();  
 }  
}

# Q2. <https://leetcode.com/problems/third-maximum-number/>

## Solution*:*

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 07/06/2023  
 \*/*public class ThirdMaximumNumber {  
 public static void main(String[] args) {  
 System.*out*.println(*findThirdMaximum*(new int[]{2, 2, 3, 1}));  
 }  
  
 static int findThirdMaximum(int[] nums) {  
 long max = Long.*MIN\_VALUE*;  
 long max2 = Long.*MIN\_VALUE*;  
 long max3 = Long.*MIN\_VALUE*;  
 for (int num : nums) {  
 if (num > max) {  
 max3 = max2;  
 max2 = max;  
 max = num;  
 } else if (num > max2 && num < max) {  
 max3 = max2;  
 max2 = num;  
 } else if (num > max3 && num < max2) {  
 max3 = num;  
 }  
 }  
 int result = (int) ((max3 == Long.*MIN\_VALUE*) ? max : max3);  
 return result;  
 }  
}

# Q3. <https://leetcode.com/problems/find-all-numbers-disappeared-in-an-array/>

## Solution:

import java.util.\*;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 07/06/2023  
 \*/*public class DisappearedNumber {  
 public static void main(String[] args) {  
 System.*out*.println(*findDisappearedNumber*(new int[]{4, 3, 2, 7, 8, 2, 3, 1}));  
 }  
  
 static List<Integer> findDisappearedNumber(int[] nums) {  
 List<Integer> result = new ArrayList<>();  
 Set<Integer> numSet = new HashSet<>();  
 for (int num : nums) {  
 numSet.add(num);  
 }  
 for (int i = 1; i <= nums.length; i++) {  
 if (!numSet.contains(i)) {  
 result.add(i);  
 }  
 }  
 return result;  
 }  
}

# Q4. <https://leetcode.com/problems/minimum-moves-to-equal-array-elements/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 07/06/2023  
 \*/*public class MinimumMovesToEqualArrayElements {  
 public static void main(String[] args) {  
 System.*out*.println(*minMoves*(new int[]{1, 2, 3}));  
 }  
  
 static int minMoves(int[] nums) {  
 int sum = 0, min = Integer.*MAX\_VALUE*;  
 for (int num : nums) {  
 sum += num;  
 min = Math.*min*(min, num);  
 }  
 return sum - (min \* nums.length);  
 }  
}

# Q5. <https://leetcode.com/problems/assign-cookies/>

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 07/06/2023  
 \*/*public class AssignCookies {  
 public static void main(String[] args) {  
 System.*out*.println(*findContentChildren*(new int[]{1, 2, 3}, new int[]{1, 2}));  
 }  
  
 static int findContentChildren(int[] g, int[] s) {  
 Arrays.*sort*(g);  
 Arrays.*sort*(s);  
 int contentChildren = 0;  
 int i = 0, j = 0;  
 while (i < g.length && j < s.length) {  
 if (s[j] >= g[i]) {  
 contentChildren++;  
 i++;  
 }  
 j++;  
 }  
 return contentChildren;  
 }  
}

# Q6. <https://leetcode.com/problems/degree-of-an-array/>

## Solution:

import java.util.HashMap;  
import java.util.Map;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 08/06/2023  
 \*/*public class DegreeOfArray {  
 public static void main(String[] args) {  
 System.*out*.println(*findShortestSubArray*(new int[]{1, 2, 2, 3, 1}));  
 }  
  
 static int findShortestSubArray(int[] nums) {  
 Map<Integer, Integer> freq = new HashMap<>();  
 int degree = Integer.*MIN\_VALUE*;  
 *//simply entering nums in hashmap, keys are numbers, values are frequency/ degree of each element  
 //Find the max degree* for (int num : nums) {  
 freq.put(num, freq.getOrDefault(num, 0) + 1);  
 degree = Math.*max*(degree, freq.get(num));  
 }  
 *//clearing the hashmap, to utilize the same map, no extra space, hence SC -> O(n)* freq.clear();  
 *//declaring two pointers* int start = 0, end = 0;  
 *// declaring current degree and minimum length of the subarray* int currDegree = Integer.*MIN\_VALUE*, minLength = Integer.*MAX\_VALUE*;  
 *//sliding window* while (end < nums.length) {  
 freq.put(nums[end], freq.getOrDefault(nums[end], 0) + 1);  
 currDegree = Math.*max*(currDegree, freq.get(nums[end]));  
 *//if currentdegree == degree, then find length of the possible shortest subarray which is minimumlength  
 //increase the start pointer value by 1 and remove the element at start index. in Map decrease the value of the number by 1* while (currDegree == degree) {  
 minLength = Math.*min*(minLength, end - start + 1);  
 freq.put(nums[start], freq.get(nums[start]) - 1);  
 if (currDegree == freq.get(nums[start]) + 1) {  
 currDegree--;  
 }  
 start++;  
 }  
 end++;  
 }  
 return minLength;  
 }  
}

# Q7. <https://leetcode.com/problems/can-place-flowers/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 08/06/2023  
 \*/*public class CanPlaceFlowers {  
 public static void main(String[] args) {  
 System.*out*.println(*canPlaceFlowers*(new int[]{1, 0, 0, 0, 1}, 1));  
 System.*out*.println(*canPlaceFlowers*(new int[]{1, 0, 0, 0, 1}, 2));  
 }  
  
 static boolean canPlaceFlowers(int[] flowerbed, int n) {  
 if (n == 0) return true;  
 for (int i = 0; i < flowerbed.length; i++) {  
 if (flowerbed[i] == 0 &&  
 (i == 0 || flowerbed[i - 1] == 0) &&  
 (i == flowerbed.length - 1 || flowerbed[i + 1] == 0)  
 ) {  
 flowerbed[i] = 1;  
 n--;  
 if (n == 0) return true;  
 }  
 }  
 return false;  
 }  
}

# Q8. <https://leetcode.com/problems/plus-one/>

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 08/06/2023  
 \*/*public class PlusOne {  
 public static void main(String[] args) {  
 System.*out*.println(Arrays.*toString*(*plusOne*(new int[]{1, 2, 3})));  
 System.*out*.println(Arrays.*toString*(*plusOne*(new int[]{1, 0, 9})));  
 }  
  
 static int[] plusOne(int[] digits) {  
 for (int i = digits.length - 1; i >= 0; i--) {  
 if (digits[i] < 9) {  
 digits[i]++;  
 return digits;  
 }  
 digits[i] = 0;  
 }  
 digits = new int[digits.length + 1];  
 digits[0] = 1;  
 return digits;  
 }  
}

# Q9. <https://leetcode.com/problems/single-number/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 08/06/2023  
 \*/*public class SingleNumber {  
 public static void main(String[] args) {  
 System.*out*.println(*singleNumber*(new int[]{4, 1, 2, 1, 2}));  
 }  
  
 static int singleNumber(int[] nums) {  
 int result = 0;  
 for (int num : nums) {  
 result ^= num;  
 }  
 return result;  
 }  
}

# Q10. <https://leetcode.com/problems/single-number-ii/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 08/06/2023  
 \*/*public class SingleNumberII {  
 public static void main(String[] args) {  
 System.*out*.println(*singleNumber*(new int[]{2, 2, 3, 2}));  
 }  
  
 static int singleNumber(int[] nums) {  
 int ones = 0;  
 int twos = 0;  
 for (int num : nums) {  
 ones = (ones ^ num) & ~twos;  
 twos = (twos ^ num) & ~ones;  
 }  
 return ones;  
 }  
}

# Q11. <https://leetcode.com/problems/single-number-ii/>

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 09/06/2023  
 \*/*public class SingleNumberIII {  
 public static void main(String[] args) {  
 System.*out*.println(Arrays.*toString*(*singleNumber*(new int[]{1, 2, 1, 3, 2, 5})));  
 }  
  
 static int[] singleNumber(int[] nums) {  
 *//find xor of two numbers* int xor = 0;  
 for (int num : nums) {  
 xor ^= num;  
 }  
 *//find the number where lowest bit = 1* int lowestBit = xor & (-xor);  
 *//group elements based on lowest bit, take and of lowest bit and the num  
 // if 0 then first number if 1, then 2nd number* int a = 0, b = 0;  
 for (int num : nums) {  
 if ((lowestBit & num) == 0) {  
 a ^= num;  
 } else {  
 b ^= num;  
 }  
 }  
 return new int[]{a, b};  
 }  
}

# Q12. <https://leetcode.com/problems/multiply-strings/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 09/06/2023  
 \*/*public class MultiplyStrings {  
 public static void main(String[] args) {  
 System.*out*.println(*multiply*("123", "456"));  
 }  
  
 static String multiply(String num1, String num2) {  
 int m = num1.length(), n = num2.length();  
 if (m == 0 || n == 0 || "0".equals(num1) || "0".equals(num2)) {  
 return "0";  
 }  
 if ("1".equals(num1)) {  
 return num2;  
 }  
 if ("1".equals(num2)) {  
 return num1;  
 }  
 int[] result = new int[m + n];  
 for (int i = m - 1; i >= 0; i--) {  
 for (int j = n - 1; j >= 0; j--) {  
 int product = (num1.charAt(i) - '0') \* (num2.charAt(j) - '0');  
  
 *//adding previous values in result array into the product* product += result[i + j + 1];  
  
 *//adding the new product into result array* result[i + j + 1] = product % 10;  
 result[i + j] += product / 10;  
 }  
 }  
 StringBuilder sb = new StringBuilder();  
 for (int element : result) {  
 if (sb.length() == 0 && element == 0) {  
 continue;  
 }  
 sb.append(element);  
 }  
 return sb.toString();  
 }  
}

# Q13.

## Solution:

# Q14.

## Solution: