PROGRAM 1

import java.util.Scanner;

public class LeadersInArray {

public static void findLeaders(int[] arr) {

int n = arr.length;

int maxFromRight = arr[n - 1];

System.out.println("Leaders in the array:");

System.out.print(maxFromRight + " ");

for (int i = n - 2; i >= 0; i--) {

if (arr[i] >= maxFromRight) {

maxFromRight = arr[i];

System.out.print(maxFromRight + " ");

}

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of elements in the array: ");

int size = scanner.nextInt();

int[] array = new int[size];

System.out.println("Enter the elements of the array:");

for (int i = 0; i < size; i++) {

array[i] = scanner.nextInt();

}

findLeaders(array);

scanner.close();

}

}

PROGRAM 2

import java.util.\*;

public class UniquePermutationsWithDuplicates {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the string: ");

String input = scanner.nextLine();

List<String> result = generateUniquePermutations(input);

System.out.println("Unique permutations in lexicographically sorted order:");

for (String perm : result) {

System.out.println(perm);

}

scanner.close();

}

public static List<String> generateUniquePermutations(String str) {

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

char[] chars = str.toCharArray();

Arrays.sort(chars);

boolean[] used = new boolean[str.length()];

StringBuilder currentPermutation = new StringBuilder();

generatePermutations(chars, used, currentPermutation, result);

return result;

}

private static void generatePermutations(char[] chars, boolean[] used, StringBuilder current, List<String> result) {

if (current.length() == chars.length) {

result.add(current.toString());

return;

}

for (int i = 0; i < chars.length; i++) {

if (used[i] || (i > 0 && chars[i] == chars[i - 1] && !used[i - 1])) {

continue;

}

used[i] = true;

current.append(chars[i]);

generatePermutations(chars, used, current, result);

used[i] = false;

current.deleteCharAt(current.length() - 1);

}

}

}

PROGRAM 3:

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class FirstNonRepeatingCharacter {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the string: ");

String input = scanner.nextLine();

char result = findFirstNonRepeatingCharacter(input);

System.out.println("First non-repeating character: " + (result != '$' ? result : "No non-repeating character"));

scanner.close();

}

public static char findFirstNonRepeatingCharacter(String str) {

Map<Character, Integer> charCountMap = new HashMap<>();

for (char ch : str.toCharArray()) {

charCountMap.put(ch, charCountMap.getOrDefault(ch, 0) + 1);

}

for (char ch : str.toCharArray()) {

if (charCountMap.get(ch) == 1) {

return ch;

}

}

return '$'; }

}

Program 4:

import java.util.Arrays;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class MinimumSwaps {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the size of the array: ");

int n = scanner.nextInt();

int[] arr = new int[n];

System.out.println("Enter the elements of the array:");

for (int i = 0; i < n; i++) {

arr[i] = scanner.nextInt();

}

int swaps = findMinimumSwaps(arr);

System.out.println("Minimum number of swaps required: " + swaps);

scanner.close();

}

public static int findMinimumSwaps(int[] arr) {

int n = arr.length;

int[] sortedArr = Arrays.copyOf(arr, n);

Arrays.sort(sortedArr);

Map<Integer, Integer> indexMap = new HashMap<>();

for (int i = 0; i < n; i++) {

indexMap.put(arr[i], i);

}

boolean[] visited = new boolean[n];

int swaps = 0;

for (int i = 0; i < n; i++) {

if (visited[i] || arr[i] == sortedArr[i]) {

continue;

}

int cycleSize = 0;

int j = i;

while (!visited[j]) {

visited[j] = true;

j = indexMap.get(sortedArr[j]);

cycleSize++;

}

if (cycleSize > 0) {

swaps += (cycleSize - 1);

}

}

return swaps;

}

}