```java
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

public class Solutions {

    private static boolean isPalindrome(String input) {
        int left = 0;
        int right = input.length() - 1;
        while(right > left) {
            if(input.charAt(left) != input.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }

    private static String longestPalindromeSubstring(String input) {
        String ans = "";
        int length = input.length();
        int low; int high;
        for(int i = 0; i<length; i++) {
            low = i;
            high = i;
            while((high + 1) < length  && input.charAt(high + 1) == input.charAt(low)) {
                high++;
            }
            while((low - 1)  >= 0 && (high + 1) < length && input.charAt(low - 1) ==
input.charAt(high + 1)) {
                low--;
                high++;
            }
            int currentLength = high - low + 1;
            int maxLength = ans.length();
            if(currentLength > maxLength) {
                ans = input.substring(low, high+1);
            }
        }
        return ans;
    }

    private static int[] merge(int[] list1, int[] list2) {
        int[] ans = new int[list1.length + list2.length];
        int current = 0;
        int point1 = 0;
        int point2 = 0;
        while(point1 < list1.length && point2 < list2.length) {
            int first = list1[point1];
            int second = list2[point2];
            if(first > second) {
                point2++;
                ans[current] = second;
            } else {
                point1++;
                ans[current] = first;
            }
            current++;
        }
        while(point1 < list1.length) {
            ans[current] = list1[point1];
            current++;
            point1++;
        }
        while(point2 < list2.length) {
            ans[current] = list2[point2];
            current++;
            point2++;
        }
        return ans;
    }

    private static boolean isomorphic(String string1, String string2) {
        if(string1.length() != string2.length()) { return false; }
        Map<Character, Character> mapping = new HashMap<>();
        for(int i = 0; i<string1.length(); i++) {
            Character first = string1.charAt(i);
            Character second = string2.charAt(i);
            if(mapping.containsKey(first)){
                if(mapping.get(first).equals(second)) { continue; }
                else { return false; }
            } else {
                if(mapping.containsValue(second)) {
                    return false;
                }
                mapping.put(first, second);
            }
        }
        return true;
    }

    private static int twoSum(int[] input, int sum) {
        Map<Integer, Integer> complements = new HashMap<>();
        int sums = 0;
        for(int element: input) {
            int complement = sum - element;
            if(complements.containsKey(complement)) {
                sums += complements.get(complement);
            }
            Integer previous = complements.getOrDefault(element, 0);
            previous++;
            complements.put(element, previous);
        }
        return sums;
    }

    private static int twoSumForThreeSum(int[] input, int sum, int avoidIndex) {
        Map<Integer, Integer> complements = new HashMap<>();
        int sums = 0;
        for(int i = 0; i<input.length; i++) {
            if(i == avoidIndex) { continue; }
            int element = input[i];
            int complement = sum - element;
            if(complements.containsKey(complement)) {
                sums += complements.get(complement);
            }
            Integer previous = complements.getOrDefault(element, 0);
            previous++;
            complements.put(element, previous);
        }
        return sums;
    }

    private static boolean threeSum(int[] input) {
        boolean completed = false;
        for(int i = 0; i<input.length; i++) {
            if(twoSumForThreeSum(input, -input[i], i) != 0) {
                completed = true;
            }
        }
        return completed;
    }

    private static String shortestSubstring(String input, String chars) {
        Map<Character, Integer> mapping = new HashMap<>();
        int inputLength = input.length();
        int totalChars = chars.length();
        int low = 0;
        int foundChars = 0;
        int minStart = 0;
        int minLength = inputLength;
        for(int i = 0; i<chars.length(); i++) {
            Character c = chars.charAt(i);
            Integer previous = mapping.getOrDefault(c, 0);
            previous++;
            mapping.put(c, previous);
        }
        for(int i = 0; i<inputLength; i++) {
            Character c = input.charAt(i);
            if(mapping.containsKey(c)){
                if(mapping.get(c) > 0) {
                    foundChars++;
                }
                Integer currentCount = mapping.get(c);
                currentCount--;
                mapping.put(c, currentCount);
            }
            while(foundChars == totalChars) {
                Character toBeRemoved = input.charAt(low);
                if(!mapping.containsKey(toBeRemoved)) { low++; continue; }
                Integer countBeforeRemoval = mapping.get(toBeRemoved);
                if(countBeforeRemoval == 0) {
                    foundChars--;
                    int currentLength = i - low + 1;
                    if(currentLength < minLength) {
                        minLength = currentLength;
                        minStart = low;
                    }
                }
                low++;
                countBeforeRemoval++;
                mapping.put(toBeRemoved, countBeforeRemoval);
            }
        }
        return input.substring(minStart, minStart + minLength);
    }

    private static String nextLargest(char[] digits) {
        int length = digits.length;
        int high = length - 1;
        int low = length - 2;
        while(low >= 0 && digits[low] > digits[high]) {
            low--;
        }
        if(low < 0) {
            return "Not Possible";
        }
        StringBuilder sb = new StringBuilder();
        for(int i = 0; i<low; i++) {
            sb.append(digits[i]);
        }
        sb.append(digits[high]);
        sb.append(digits[low]);
        for(int i = high-1; i>low; i--) {
            sb.append(digits[i]);
        }
        return sb.toString();
    }

    private static void testIsPalindrome() {
        String[] inputs = new String[]{"Test", "tacocat", "taco cat", "naan", "not actually"};
        for(String input: inputs) {
            System.out.println(input + " - " + isPalindrome(input));
        }
    }

    private static void testLongestPalindromeSubstring() {
        String[] inputs = new String[]{"iwentskiiksnotreally", "aaaaaa", "tacocat", "randomstring",
"racecar"};
        for(String input: inputs) {
            System.out.println(input + " - " + longestPalindromeSubstring(input));
        }
    }

    private static void testMerge() {
        int[] arr1 = {1, 3};
        int[] arr2 = {2, 4, 6, 8};
        System.out.println(Arrays.toString(arr1) + " and " + Arrays.toString(arr2) + " yield: " +
                Arrays.toString(merge(arr1, arr2)));
    }

    private static void testIsomorphic() {
        String[] input1 = new String[]{"aag", "adg", "avd", "vds", "aag"};
        String[] input2 = new String[]{"vvc", "aag", "tad", "vdv", "adg"};
        for(int i = 0; i<input1.length; i++) {
            System.out.println(input1[i] + " and " + input2[i] + ": " + isomorphic(input1[i],
input2[i]));
        }
    }

    private static void testTwoSum() {
        int[] arr = new int[]{1, 5, 7, -1};
        int sum = 13;
        System.out.println(Arrays.toString(arr) + " 2sum " + sum + ": " + twoSum(arr, sum));
    }

    private static void testThreeSum() {
        int[] arr = new int[]{-1, 0, 1, 2, -1, -4};
        System.out.println(Arrays.toString(arr) + " 3sum " + ": " + threeSum(arr));
    }

    private static void testShortestSubstring() {
        String[] inputs = new String[]{"ADOBECODEBANC", "geeksforgeeks", "tacocat", "randomstring",
"racecar"};
        String[] chars = new String[]{"ABC", "ork", "tacocat", "randomstring", "racecar"};
        for(int i = 0; i<inputs.length; i++) {
            System.out.println(inputs[i] + " and " + chars[i] + " - " +
shortestSubstring(inputs[i], chars[i]));
        }
    }

    private static void testNextLargest() {
        String[] inputs = new String[]{"1234", "4321", "218765", "534976"};
        for(String input: inputs) {
            System.out.println(input + " - " + nextLargest(input.toCharArray()));
        }
    }

    public static void main(String[] args) {
//        testIsPalindrome();
//        testLongestPalindromeSubstring();
//        testMerge();
//        testIsomorphic();
//        testTwoSum();
//        testThreeSum();
//        testShortestSubstring();
//        testNextLargest();
    }
}
```