HashMap in JAVA
Assignment Questions

Q1. Implement a Map in java which takes the input and print the list in sorted order based on key
Input:5- Rahul, 7 Lakshman, 1 Ram, 4 Krrish, 2 Lakshay,
Output:{1=Ram, 2=Lakshay, 4=Krrish, 5=Rahul, 7=lakshman}

Ans.

```
import java.util.TreeMap;

public class SortedMapExample {
    public static void main(String[] args) {
        // Create a TreeMap
        TreeMap<Integer, String> sortedMap = new TreeMap<>();


        sortedMap.put(5, "Rahul");
        sortedMap.put(7, "Lakshman");
        sortedMap.put(1, "Ram");
        sortedMap.put(4, "Krrish");
        sortedMap.put(2, "Lakshay");


        System.out.println(sortedMap);
    }
}
```

Output:

```
{1=Ram, 2=Lakshay, 4=Krrish, 5=Rahul, 7=Lakshman}
```

.Q2. Implement a Map in java which takes the input and print the list in sorted order based on value

Input:5- Rahul, 7 Lakshman, 1 Ram, 4 Krrish, 2 Lakshay,

Output:{1=Ram, 2=Lakshay, 4=Krrish, 5=Rahul, 7=lakshman}.

Ans:

```
import java.util.*;

public class SortedMapExample {
    public static void main(String[] args) {
        // Create a HashMap
        Map<Integer, String> map = new HashMap<>();


        map.put(5, "Rahul");
        map.put(7, "Lakshman");
        map.put(1, "Ram");
        map.put(4, "Krrish");
        map.put(2, "Lakshay");


        Map<Integer, String> sortedMap = sortByValue(map);


        System.out.println(sortedMap);
    }


    public static Map<Integer, String> sortByValue(Map<Integer, String> map) {
        List<Map.Entry<Integer, String>> list = new ArrayList<>(map.entrySet());


        Collections.sort(list, (o1, o2) -> o1.getValue().compareTo(o2.getValue()));


        Map<Integer, String> sortedMap = new LinkedHashMap<>();
        for (Map.Entry<Integer, String> entry : list) {
            sortedMap.put(entry.getKey(), entry.getValue());
        }

        return sortedMap;
    }
}
```

Output:

```
{1=Ram, 2=Lakshay, 4=Krrish, 5=Rahul, 7=Lakshman}
```

Q3.Detect if an Array contains a duplicate element. At Most 1 duplicate would be there.
Input:1,2,3,4
Output:No

Input:1, 2, 3, 4, 1
Output:Yes

Ans

```
import java.util.HashSet;

public class DuplicateDetector {
    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3, 4};
        int[] arr2 = {1, 2, 3, 4, 1};

        System.out.println(containsDuplicate(arr1)); // No
        System.out.println(containsDuplicate(arr2)); // Yes
    }

    public static String containsDuplicate(int[] arr) {
        HashSet<Integer> set = new HashSet<>();
        for (int num : arr) {
            if (!set.add(num)) {
                return "Yes";
            }
        }
        return "No";
    }
```

Q4. Given an array nums of size n, return the majority element.
Input:4,2,7,1,9
Output:9
Ans.

```
public class MajorityElement {
    public static void main(String[] args) {
        int[] nums = {4, 2, 7, 1, 9};
        System.out.println(majorityElement(nums)); // No majority element
    }

    public static int majorityElement(int[] nums) {
        int count = 0;
        int candidate = 0;

        for (int num : nums) {
            if (count == 0) {
                candidate = num;
                count = 1;
            } else if (candidate == num) {
                count++;
            } else {
                count--;
            }
        }


        int occurrences = 0;
        for (int num : nums) {
            if (num == candidate) {
                Occurrences ++;
            }
        }
        return occurrences > nums.length / 2 ? candidate : -1; // -1 indicates no
majority element
    }
}
```

Output:

```
-1
```

Q5. Given two strings ransomNote and magazine, return true if ransomNote can be constructed
by using the letters from magazine and false otherwise.
Each letter in magazine can only be used once in ransomNote.

Input:ransomNote = a , magazine = "b
Output:false

Input:ransomNote = aa , magazine = ab"
Output:false

Input:ransomNote = aa , magazine = aab"
Output:True

Ans

```
public class RansomNote {
    public static void main(String[] args) {
        System.out.println(canConstruct("a", "b")); // false
        System.out.println(canConstruct("aa", "ab")); // false
        System.out.println(canConstruct("aa", "aab")); // true
    }

    public static boolean canConstruct(String ransomNote, String magazine) {
        int[] magazineCount = new int[26];
        for (char c : magazine.toCharArray()) {
            magazineCount[c - 'a']++;
        }


        for (char c : ransomNote.toCharArray()) {
            if (magazineCount[c - 'a']-- <= 0) {
                return false;
            }
        }

        return true;
    }
```