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:

In Java, we can use the `TreeMap` class to implement a map that automatically sorts its keys. Here's an example:

```
import java.util.*;

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

        // Adding elements to the TreeMap
        sortedMap.put(3, "Three");
        sortedMap.put(1, "One");
        sortedMap.put(2, "Two");
        sortedMap.put(4, "Four");

        // Printing the sorted map based on keys
        System.out.println("Sorted Map based on keys:");
        for (Map.Entry<Integer, String> entry : sortedMap.entrySet()) {
            System.out.println("Key: " + entry.getKey() + ", Value: " + entry.getValue());
        }
    }
}
```

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: {Rahul=5, krrish=4, lakshay=7, lakshman=2, ram=1}

Ans:

To create a `Map` in Java that sorts based on values, you can use a `List` to hold the entries of the map and then sort that list based on the values. Here's an example:

```
import java.util.*;

public class SortedMapByValueExample {
    public static void main(String[] args) {
        // Creating a HashMap
        HashMap<String, Integer> unsortedMap = new HashMap<>();

        // Adding elements to the HashMap
        unsortedMap.put("Three", 3);
        unsortedMap.put("One", 1);
        unsortedMap.put("Two", 2);
        unsortedMap.put("Four", 4);

        // Creating a list from the entries of the HashMap
        List<Map.Entry<String, Integer>> entryList = new ArrayList<>(unsortedMap.entrySet());

        // Sorting the list based on values
        Collections.sort(entryList, Map.Entry.comparingByValue());

        // Printing the sorted list based on values
        System.out.println("Sorted Map based on values:");
        for (Map.Entry<String, Integer> entry : entryList) {
            System.out.println("Key: " + entry.getKey() + ", Value: " + entry.getValue());
        }
    }
}
```

3)

Q3.Detect if an Array contains a duplicate element. At Most 1 duplicate would be there.

Input: 1,2,3,4
Output: No

Ans:

To detect if an array contains at most one duplicate element, we can use a HashSet to keep track of the elements you have encountered so far. Here's a simple Java example:

```
import java.util.HashSet;

public class DuplicateDetection {
    public static boolean containsDuplicate(int[] nums) {
```

```
        HashSet<Integer> set = new HashSet<>();

        for (int num : nums) {
            // If the set already contains the element, it's a duplicate
            if (!set.add(num)) {
                return true;
            }
        }

        return false;
    }

    public static void main(String[] args) {
        // Example usage
        int[] array1 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; // No duplicates
        int[] array2 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10}; // One duplicate

        System.out.println("Array 1 contains duplicate: " + containsDuplicate(array1));
        System.out.println("Array 2 contains duplicate: " + containsDuplicate(array2));
    }
}
```

Q4. Given an array nums of size n, return the majority element.
Input:
output: 9

Ans:

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

public class MajorityElement {
    public static int majorityElement(int[] nums) {
        Map<Integer, Integer> countMap = new HashMap<>();

        for (int num : nums) {
            countMap.put(num, countMap.getOrDefault(num, 0) + 1);
        }

        int majorityElement = 0;
        int majorityCount = 0;

        for (Map.Entry<Integer, Integer> entry : countMap.entrySet()) {
```

```
            if (entry.getValue() > majorityCount) {
                majorityCount = entry.getValue();
                majorityElement = entry.getKey();
            }
        }

        return majorityElement;
    }

    public static void main(String[] args) {
        // Example usage
        int[] nums = {2, 2, 1, 1, 1, 2, 2};

        int majorityElement = majorityElement(nums);
        System.out.println("Majority Element: " + majorityElement);
    }
}
```

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

Ans:

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

public class RansomNote {
    public static boolean canConstruct(String ransomNote, String magazine) {
        Map<Character, Integer> charCount = new HashMap<>();

        // Count occurrences of each character in magazine
        for (char c : magazine.toCharArray()) {
            charCount.put(c, charCount.getOrDefault(c, 0) + 1);
        }

        // Check if ransomNote can be constructed
        for (char c : ransomNote.toCharArray()) {
            if (!charCount.containsKey(c) || charCount.get(c) <= 0) {
                return false;
```

```
            }
            charCount.put(c, charCount.get(c) - 1);
        }

        return true;
    }

    public static void main(String[] args) {
        // Example usage
        String ransomNote1 = "aabb";
        String magazine1 = "aabbc";
        System.out.println(canConstruct(ransomNote1, magazine1)); // Should print true

        String ransomNote2 = "aa";
        String magazine2 = "aab";
        System.out.println(canConstruct(ransomNote2, magazine2)); // Should print false
    }
}
```