**Assignment HashMap**

Q1. Q1. Implement a Map in java which takes the input and print the list in sorted order based on key.

Input:

Output:

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

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

import java.util.Map;

import java.util.TreeMap;

public class SortedMapExample {

public static void main(String[] args) {

// Create a TreeMap to store and sort entries by key

Map<Integer, String> map = new TreeMap<>();

// Add entries to the map

map.put(5, "Rahul");

map.put(7, "Lakshman");

map.put(1, "Ram");

map.put(4, "Krrish");

map.put(2, "Lakshay");

// Print the map entries in sorted order by key

System.out.println(map);

}

}

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

Input:

Output:

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

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

import java.util.\*;

public class SortMapByValue {

// Method to sort a map by its values

public static Map<String, Integer> sortByValue(Map<Integer, String> map) {

// Convert the map entries to a list

List<Map.Entry<Integer, String>> entryList = new ArrayList<>(map.entrySet());

// Sort the list based on the values of the map entries

entryList.sort(Map.Entry.comparingByValue());

// Create a LinkedHashMap to maintain the order of the sorted entries

Map<String, Integer> sortedMap = new LinkedHashMap<>();

// Populate the LinkedHashMap with sorted entries

for (Map.Entry<Integer, String> entry : entryList) {

sortedMap.put(entry.getValue(), entry.getKey());

}

return sortedMap;

}

public static void main(String[] args) {

// Create a map with integer keys and string values

Map<Integer, String> map = new TreeMap<>();

map.put(5, "Rahul");

map.put(7, "Lakshman");

map.put(1, "Ram");

map.put(4, "Krrish");

map.put(2, "Lakshay");

// Sort the map by its values

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

// Print the sorted map

System.out.println(sortedMap);

}

}

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

Input:

Output:

Input:

Output:

1,2,3,4

No

1, 2, 3, 4, 1

Yes.

import java.util.HashSet;

import java.util.Set;

public class DuplicateDetector {

// Method to check if the array contains at most one duplicate

public static boolean containsAtMostOneDuplicate(int[] array) {

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

int duplicateCount = 0;

for (int value : array) {

if (!set.add(value)) {

duplicateCount++;

if (duplicateCount > 1) {

return true; // More than one duplicate found

}

}

}

return duplicateCount == 1; // At most one duplicate found

}

public static void main(String[] args) {

int[] array1 = {1, 2, 3, 4};

int[] array2 = {1, 2, 3, 4, 1};

System.out.println("Array 1 contains at most one duplicate: " + (containsAtMostOneDuplicate(array1) ? "Yes" : "No"));

System.out.println("Array 2 contains at most one duplicate: " + (containsAtMostOneDuplicate(array2) ? "Yes" : "No"));

}

}

Q4. Given an array nums of size n, return the majority element.

Input:

Output:

4,2,7,1,9

9

public class MajorityElementFinder {

// Method to find the majority element using Boyer-Moore Voting Algorithm

public static int findMajorityElement(int[] nums) {

int candidate = nums[0];

int count = 1;

// Phase 1: Find the candidate

for (int i = 1; i < nums.length; i++) {

if (nums[i] == candidate) {

count++;

} else {

count--;

if (count == 0) {

candidate = nums[i];

count = 1;

}

}

}

// Phase 2: Verify the candidate

count = 0;

for (int num : nums) {

if (num == candidate) {

count++;

}

}

return count > nums.length / 2 ? candidate : -1; // Return -1 if no majority element

}

public static void main(String[] args) {

int[] nums1 = {4, 2, 7, 1, 9};

int[] nums2 = {3, 3, 4, 2, 4, 4, 2, 4, 4};

System.out.println("Majority element in array 1: " + findMajorityElement(nums1));

System.out.println("Majority element in array 2: " + findMajorityElement(nums2));

}

}

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.

import java.util.HashMap;

import java.util.Map;

public class RansomNoteConstructor {

// Method to check if the ransomNote can be constructed from magazine

public static boolean canConstruct(String ransomNote, String magazine) {

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

// Count characters in magazine

for (char c : magazine.toCharArray()) {

magazineCount.put(c, magazineCount.getOrDefault(c, 0) + 1);

}

// Check if ransomNote can be constructed

for (char c : ransomNote.toCharArray()) {

if (magazineCount.getOrDefault(c, 0) <= 0) {

return false; // Not enough characters in magazine

}

magazineCount.put(c, magazineCount.get(c) - 1);

}

return true; // All characters were available in the magazine

}

public static void main(String[] args) {

String ransomNote1 = "a";

String magazine1 = "b";

System.out.println("Can construct ransomNote1: " + canConstruct(ransomNote1, magazine1));

String ransomNote2 = "aa";

String magazine2 = "ab";

System.out.println("Can construct ransomNote2: " + canConstruct(ransomNote2, magazine2));

String ransomNote3 = "aa";

String magazine3 = "aab";

System.out.println("Can construct ransomNote3: " + canConstruct(ransomNote3, magazine3));

}

}