Rajalakshmi Engineering College

Name: Saranya Vimalanathan
Email: 240701620@rajalakshmi.edu.in
Roll no: 240701620
Phone: 9789689339
Branch: REC
Department: CSE - Section 9
Batch: 2028
Degree: B.E - CSE

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 10_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 20

Section 1 : COD

1. Problem Statement

Arjun is working on a program that checks if one set of numbers is a subset of another. If Set B is a subset of Set A, the program should print "YES" followed by the sorted elements of Set B. If Set B is not a subset of Set A, the program should print "NO" followed by the average of all elements from both sets combined, rounded to two decimal places.

Implement a class Solution with the required method to perform the subset check using TreeSet in Java.

Input Format

The first line contains an integer n - the number of elements in Set A.

The second line contains n space-separated integers - the elements of Set A.

The third line contains an integer m - the number of elements in Set B.

The fourth line contains m space-separated integers - the elements of Set B.

Output Format

If Set B is a subset of Set A, print "YES" followed by the sorted values of Set B.

Otherwise, print "NO" followed by the average of all numbers in both sets (rounded to two decimal places).

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5
3
2 3 5
Output: YES 2 3 5

Answer

```
import java.util.*;

class Solution {
   public static void checkSubset(TreeSet<Integer> setA, TreeSet<Integer> setB,
int totalCount, long totalSum) {
        if (setA.containsAll(setB)) {
            System.out.print("YES ");
            for (int num : setB) {
                System.out.print(num + " ");
            }
        } else {
            ArrayList<Integer> all = new ArrayList<>();
            all.addAll(setA);
            all.addAll(setB);
            double sum = 0;
            for (int x : all) sum += x;
            double avg = sum / all.size();
            System.out.printf("NO %.2f", avg);
        }
    }
}
class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        TreeSet<Integer> setA = new TreeSet<>();
        long sum = 0;
        for (int i = 0; i < n; i++) {
            int num = sc.nextInt();
            setA.add(num);
            sum += num;
        }
        int m = sc.nextInt();
        TreeSet<Integer> setB = new TreeSet<>();
        for (int i = 0; i < m; i++) {
            int num = sc.nextInt();
            setB.add(num);
            sum += num;
        }
        Solution.checkSubset(setA, setB, n + m, sum);
        sc.close();
    }
}
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

David is managing an employee database where each employee has a unique ID, name, and department. He wants to ensure that duplicate employee IDs are not added to the system. Implement a Java program that allows adding employees to the system, displaying all employees, and checking if an employee exists based on the given ID.

Implement a class EmployeeDatabase that contains a HashSet to store employee records. The Employee class should be a user-defined object containing employee details. The main class should handle user operations and interact with the EmployeeDatabase class.

Input Format

The first line contains an integer n representing the number of employees to be added.

The next n lines follow, each containing:

1. An integer employee_id
2. A string name
3. A string department

The next line contains an integer m representing the number of queries.

The next m lines follow, each containing an employee ID to check for existence.

Output Format

The output prints a list of all employees added in the format:

"ID: <employee_id>, Name: <name>, Department: <department>"

For each query, output "Employee exists" if the ID is found, otherwise "Employee not found".

Refer to the sample output for formatting specifications.

Sample Test Case

```
Input: 3
101 John IT
102 Alice HR
103 Bob Finance
2
101
104
Output: ID: 101, Name: John, Department: IT
ID: 102, Name: Alice, Department: HR
ID: 103, Name: Bob, Department: Finance
Employee exists
Employee not found
```

Answer

```
import java.util.*;

// You are using Java
class Employee {
   HashSet
}

class EmployeeDatabase {
   //Type your code here
}

class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      EmployeeDatabase db = new EmployeeDatabase();
      int n = sc.nextInt();
      for (int i = 0; i < n; i++) {
         int id = sc.nextInt();
         String name = sc.next();
         String department = sc.next();
         db.addEmployee(id, name, department);
      }
      db.displayEmployees();
      int m = sc.nextInt();
      for (int i = 0; i < m; i++) {
         int id = sc.nextInt();
         if (db.checkEmployee(id))
            System.out.println("Employee exists");
         else
            System.out.println("Employee not found");
      }
      sc.close();
   }
}
```

Status : Wrong ***Marks : 0/10***

3. Problem Statement

Aryan is developing a voting system for a college election. Each vote is recorded as an entry in an array, where every student's vote is represented by a candidate's ID. Since it's a majority-rule election, the winner is the candidate who receives more than n/2 votes, where n is the total number of votes cast.

To quickly determine the winner, Aryan decides to use a HashMap to count the occurrences of each vote and identify the candidate who has received more than half of the total votes.

Example

Input

7

2 2 1 2 2 2 3

Output

2

Explanation

The votes are: 2, 2, 1, 2, 2, 3, 2

Count of each candidate:

2 appears 5 times1 appears once3 appears once

The majority element is the one that appears more than N/2 times. Since 7/2 = 3.5, a number must appear at least 4 times to be the majority.

The number 2 appears 5 times, which is greater than 3.5, so the output is 2.

Input Format

The first line contains an integer N representing the number of votes cast.

The second line contains N space-separated integers representing the votes, where each integer corresponds to a candidate.

Output Format

The output prints an integer representing the majority element (the candidate who received more than N/2 votes).

If no such candidate exists, print -1.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
2 2 1 2 2 2 3

Output: 2

Answer

```
import java.util.HashMap;
import java.util.Scanner;

// You are using Java
class MajorityElementFinder {
   //type yoimport java.util.*;

class Student implements Comparable<Student> { HashMap
   int rollNo;
   String name;
   int attendance;

   Student(int rollNo, String name) {
      this.rollNo = rollNo;
      this.name = name;
      this.attendance = 0;
   }

   public int compareTo(Student s) {
      return this.rollNo - s.rollNo;
   }

   public boolean equals(Object o) {
      if (this == o) return true;
      if (!(o instanceof Student)) return false;
      Student s = (Student) o;
      return this.rollNo == s.rollNo;
   }
ur code here
}
class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      int N = scanner.nextInt();
      int[] arr = new int[N];

      for (int i = 0; i < N; i++) {
          arr[i] = scanner.nextInt();
      }

      int result = MajorityElementFinder.findMajorityElement(arr);
      System.out.println(result);

      scanner.close();
   }
}
```

Status : Wrong ***Marks : 0/10***

4. Problem Statement

A college professor wants to keep track of students who attend classes. Each student has a unique roll number and their attendance count increases every time they attend a class. The system should allow adding a student, marking their attendance, and displaying all students with their total attendance.

Your task is to implement a Java program using TreeSet to maintain students in sorted order of roll numbers and track their attendance count.

Operations:

A roll_no name   Add a student with roll number and name (if not already added).M roll_no   Mark attendance for the student with the given roll number (increase their count by 1).D   Display all students in ascending order of roll number along with their attendance count.

Input Format

The first line contains an integer N - the number of students.

The next N lines contain one of the following commands:

A roll_no name

M roll_no

D

- A (Add)   Adds a new student with a unique roll number and name.
- M (Mark)   Increases attendance count for the given roll number.
- D (Display)   Prints all students in ascending order of roll number.

Output Format

For D, output prints each student's roll number, name, and attendance count in ascending order of roll number.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
A 101 Alice
A 102 Bob
M 101
M 101
D
Output: 101 Alice 2
102 Bob 0

Answer

```
// You are using Java
import java.util.*;

class Student implements Comparable<Student> {
   int rollNo;
   String name;
   int attendance;

   public Student(int rollNo, String name) {
       this.rollNo = rollNo;
       this.name = name;
       this.attendance = 0;
   }

   public void markAttendance() {
       this.attendance++;
   }

   @Override
   public int compareTo(Student other) {
       return Integer.compare(this.rollNo, other.rollNo);
   }

   @Override
   public boolean equals(Object obj) {
       if (this == obj) return true;
       if (!(obj instanceof Student)) return false;
       Student other = (Student) obj;
       return this.rollNo == other.rollNo;
   }

   @Override
   public int hashCode() {
       return Objects.hash(rollNo);
   }
}

public class Main {
   public static void main(String[] args) {
       Scanner scanner = new Scanner(System.in);
       int N = Integer.parseInt(scanner.nextLine());

       TreeSet<Student> students = new TreeSet<>();

       for (int i = 0; i < N; i++) {
           String line = scanner.nextLine();
           String[] parts = line.split(" ");

           if (parts[0].equals("A")) {
               int rollNo = Integer.parseInt(parts[1]);
               String name = parts[2];
               Student newStudent = new Student(rollNo, name);
               students.add(newStudent);
           } else if (parts[0].equals("M")) {
               int rollNo = Integer.parseInt(parts[1]);
               for (Student s : students) {
                   if (s.rollNo == rollNo) {
                       s.markAttendance();
                       break;
                   }
               }
           } else if (parts[0].equals("D")) {
               for (Student s : students) {
                   System.out.println(s.rollNo + " " + s.name + " " + s.attendance);
               }
           }
       }
   }
}
```

Status : Correct ***Marks : 10/10***