Rajalakshmi Engineering College

Name: Prajeet V
Email: 240701389@rajalakshmi.edu.in
Roll no: 240701389
Phone: 9363389322
Branch: REC
Department: CSE - Section 8
Batch: 2028
Degree: B.E - CSE

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 10_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : COD

1. 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

```
import java.util.*;

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

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

    public void markAttendance() {
        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 == null || getClass() != obj.getClass()) return false;
        Student s = (Student) obj;
        return rollNo == s.rollNo;
    }

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

    @Override
    public String toString() {
        return rollNo + " " + name + " " + attendance;
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        TreeSet<Student> students = new TreeSet<>();

        for (int i = 0; i < n; i++) {
```

```java
            String command = sc.next();

            if (command.equals("A")) {
                int rollNo = sc.nextInt();
                String name = sc.next();
                students.add(new Student(rollNo, name));
            }
            else if (command.equals("M")) {
                int rollNo = sc.nextInt();
                for (Student s : students) {
                    if (s.rollNo == rollNo) {
                        s.markAttendance();
                        break;
                    }
                }
            }
            else if (command.equals("D")) {
                for (Student s : students) {
                    System.out.print(s + " ");
                }
            }
        }

        sc.close();
    }
}
// You are using Java
```

2. 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

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

class MajorityElementFinder {
    public static int findMajorityElement(int[] arr) {
        HashMap<Integer, Integer> countMap = new HashMap<>();
        int n = arr.length;
        int threshold = n / 2;

        for (int vote : arr) {
            int count = countMap.getOrDefault(vote, 0) + 1;
            countMap.put(vote, count);
            if (count > threshold) {
                return vote; // Found majority
            }
        }

        return -1; // No majority element
    }
}
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();
    }
}
```

3.   Problem Statement

The city library maintains a record of books available for lending. Each book is uniquely identified by its ISBN number, along with its title and author. The librarian wants to efficiently store and manage these records, ensuring books can be listed in the order they were added.

Your task is to implement a Library Management System using HashSet where:

The librarian adds books with ISBN, title, and author.The librarian can remove books by providing an ISBN.Finally, the librarian displays the available books in the order they were added.

Implement a class Library that will handle these operations. The main function should manage user input and interact with the Library class accordingly.

Input Format

The first line contains an integer n – the number of books to be added.

The next n lines contain three values: ISBN (integer), Title (string without spaces), and Author (string without spaces).

1. An integer employee_id
2. A string title
3. A string author name

The next line contains an integer m – the number of books to be removed.

The next m lines follow, each contains an ISBN number to remove.

Output Format

The output prints a list of books available in the library after performing all operations in the format:

"ISBN: <isbn>, Title: <title>, Author: <author>"

If no books remain, print: "No books available"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1234 JavaCompleteGuide JohnDoe
5678 PythonBasics JaneDoe
9012 DataStructures AliceSmith
1
5679
Output: ISBN: 1234, Title: JavaCompleteGuide, Author: JohnDoe
ISBN: 9012, Title: DataStructures, Author: AliceSmith
ISBN: 5678, Title: PythonBasics, Author: JaneDoe

Answer

```java
import java.util.*;

class Book {
    int isbn;
    String title;
    String author;

    public Book(int isbn, String title, String author) {
        this.isbn = isbn;
        this.title = title;
        this.author = author;
    }

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

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Book other = (Book) obj;
```

```java
            return this.isbn == other.isbn;
    }

    @Override
    public String toString() {
        return "ISBN: " + isbn + ", Title: " + title + ", Author: " + author;
    }
}

class Library {
    LinkedHashSet<Book> books = new LinkedHashSet<>();

    public void addBook(int isbn, String title, String author) {
        Book b = new Book(isbn, title, author);
        books.add(b);
    }

    public void removeBook(int isbn) {
        books.removeIf(book -> book.isbn == isbn);
    }

    public void displayBooks() {
        if (books.isEmpty()) {
            System.out.print("No books available");
        } else {
            for (Book b : books) {
                System.out.println(b);
            }
        }
    }
}

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Library library = new Library();
        int n = sc.nextInt();
        for (int i = 0; i < n; i++) {
            int isbn = sc.nextInt();
            String title = sc.next();
            String author = sc.next();
            library.addBook(isbn, title, author);
        }
```

```
    int m = sc.nextInt();
    for (int i = 0; i < m; i++) {
        int isbn = sc.nextInt();
        library.removeBook(isbn);
    }
    library.displayBooks();
    sc.close();
  }
}
```

Status : Correct   *Marks : 10/10*

4. 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

```java
import java.util.*;

class Solution {
    public static void checkSubset(TreeSet<Integer> setA, TreeSet<Integer> setB,
int totalCount, long sum) {
        if (setA.containsAll(setB)) {
            System.out.print("YES ");
            for (int num : setB) {
                System.out.print(num + " ");
            }
        } else {
            double avg = (double) sum / totalCount;
            System.out.print("NO " + String.format("%.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*