Rajalakshmi Engineering College

Name: Rahul G
Email: 241901087@rajalakshmi.edu.in
Roll no: 241901087
Phone: 8248227615
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 10_MCQ

Attempt : 1
Total Mark : 15
Marks Obtained : 14

Section 1 : MCQ

1. Which statement is true about HashSet and TreeSet?

Answer

TreeSet provides sorted elements

Status : Correct ***Marks : 1/1***

2. Which of the following is true about TreeMap?

Answer

It maintains natural ordering

Status : Correct ***Marks : 1/1***

3. What will happen if you add elements in descending order in a TreeSet?

Answer

They are sorted in ascending order

Status : Correct ***Marks : 1/1***

4. How does HashSet check for duplicate elements?

Answer

Using equals() and hashCode()

Status : Correct ***Marks : 1/1***

5. What will be the output of the following code?

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        HashMap<String, Integer> map = new HashMap<>();
        map.put("X", 10);
        map.put("Y", 20);
        map.put("Z", 30);
        map.remove("Y");
        System.out.println(map);
    }
}
```

Answer

{X=20, Z=30}

Status : Wrong ***Marks : 0/1***

6. Which of the following allows null keys in Java?

Answer

HashMap

Status : Correct ***Marks : 1/1***

7. Which method retrieves the lowest key in a TreeMap?

Answer

firstKey()

Status : Correct ***Marks : 1/1***

8. What happens if two keys have the same hash code in a HashMap?

Answer

A linked list is used to store values with the same hash

Status : Correct ***Marks : 1/1***

9. What happens when you add duplicate elements to a HashSet?

Answer

The duplicate is ignored

Status : Correct ***Marks : 1/1***

10. What is the time complexity of retrieving an element from a HashSet?

Answer

O(1)

Status : Correct ***Marks : 1/1***

11. What will be the output of the following code?

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        HashMap<String, Integer> map = new HashMap<>();
```

```
        map.put("A", 1);
        map.put("B", 2);
        map.put("C", 3);
        System.out.println(map.containsKey("B"));
    }
}
```

Answer

true

Status : Correct ***Marks : 1/1***

12. Which of the following is true about HashMap?

Answer

It is not synchronized

Status : Correct ***Marks : 1/1***

13. What will happen if you add a null element to a TreeSet?

Answer

An exception occurs

Status : Correct ***Marks : 1/1***

14. Which method removes all elements from a Set?

Answer

clear()

Status : Correct ***Marks : 1/1***

15. What will be the output of the following code?

```
import java.util.*;
class Main {
```

```
    public static void main(String[] args) {
        HashMap<String, String> map = new HashMap<>();
        map.put("A", "Apple");
        map.put("B", "Banana");
        map.put("C", "Cherry");
        map.replace("B", "Blueberry");
        System.out.println(map);
    }
}
```

Answer

{A=Apple, B=Blueberry, C=Cherry}

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Rahul G
Email: 241901087@rajalakshmi.edu.in
Roll no: 241901087
Phone: 8248227615
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 10_Q1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : COD

1. Problem Statement

A city traffic management system needs to track vehicles entering a toll booth. Each vehicle is uniquely identified by its registration number. The system should allow adding vehicles to a record, ensuring that no duplicate registration numbers exist. The vehicles should be stored in a HashSet, which does not guarantee any specific order.

Your task is to implement a program using a HashSet that allows adding vehicle details and displaying the records.

Input Format

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

The next N lines contain details of each vehicle in the format: "RegNumber

OwnerName VehicleType"

1. RegNumber (String) - A unique registration number (Alphanumeric).
2. OwnerName (String) - The name of the vehicle owner.
3. VehicleType (String, Car, Bike, or Truck) - The type of vehicle.

If a vehicle with the same registration number is already present, ignore the duplicate entry.

Output Format

The output prints the unique vehicle records in any order (since HashSet does not maintain order).

Output format: "RegNumber OwnerName VehicleType"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
KA01AB1234 John Car
MH02CD5678 Alice Bike
DL03EF9012 Bob Truck
TN04GH3456 Mike Car
KA01AB1234 John Car
Output: TN04GH3456 Mike Car
KA01AB1234 John Car
MH02CD5678 Alice Bike
DL03EF9012 Bob Truck

Answer

```
import java.util.*;

class Vehicle {
   String regNumber, ownerName, vehicleType;

   Vehicle(String regNumber, String ownerName, String vehicleType) {
      this.regNumber = regNumber;
      this.ownerName = ownerName;
      this.vehicleType = vehicleType;
```

```
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Vehicle)) return false;
        Vehicle v = (Vehicle) o;
        return regNumber.equals(v.regNumber);
    }

    @Override
    public int hashCode() {
        return regNumber.hashCode();
    }

    @Override
    public String toString() {
        return regNumber + " " + ownerName + " " + vehicleType;
    }
}

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

        for (int i = 0; i < n; i++) {
            String regNumber = sc.next();
            String ownerName = sc.next();
            String vehicleType = sc.next();
            vehicles.add(new Vehicle(regNumber, ownerName, vehicleType));
        }

        for (Vehicle v : vehicles) {
            System.out.println(v);
        }

        sc.close();
    }
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Rahul G
Email: 241901087@rajalakshmi.edu.in
Roll no: 241901087
Phone: 8248227615
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 10_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : COD

1. Problem Statement

John is organizing a fruit festival, and the quantities of various fruits are stored in a HashMap where fruit names are keys and quantities are values.

Help him develop a program to find the total quantity of fruits for the festival by summing up the values in the HashMap.

Input Format

The input consists of fruit quantities in the format 'fruitName:quantity', where fruitName is the name of the fruit(a string), and quantity is a double value representing the quantity.

The input is terminated by entering "done".

Output Format

The output prints a double value, representing the sum of values in the HashMap, rounded off to two decimal places.

If the value is not numeric, print "Invalid input".

If any special characters other than ':' are entered, print "Invalid format".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Banana:15.2
Orange:56.3
Mango:47.3
done

Output: 118.80

Answer

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

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        HashMap<String, Double> fruits = new HashMap<>();
        double total = 0.0;
        boolean invalidInput = false;
        boolean invalidFormat = false;

        while (true) {
            String input = sc.nextLine().trim();
            if (input.equalsIgnoreCase("done")) {
                break;
            }

            // Check format: must contain exactly one colon
            if (!input.contains(":") || input.startsWith(":") || input.endsWith(":") ||
input.split(":").length != 2) {
                invalidFormat = true;
                break;
```

```
            }

            String[] parts = input.split(":");
            String fruit = parts[0];
            String qtyStr = parts[1];

            // Check for invalid characters (anything other than letters in fruit name
and ':' separator)
            if (!fruit.matches("[A-Za-z]+")) {
                invalidFormat = true;
                break;
            }

            try {
                double quantity = Double.parseDouble(qtyStr);
                fruits.put(fruit, quantity);
            } catch (NumberFormatException e) {
                invalidInput = true;
                break;
            }
        }

        if (invalidFormat) {
            System.out.println("Invalid format");
        } else if (invalidInput) {
            System.out.println("Invalid input");
        } else {
            for (double val : fruits.values()) {
                total += val;
            }
            System.out.printf("%.2f\n", total);
        }

        sc.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Rahul G
Email: 241901087@rajalakshmi.edu.in
Roll no: 241901087
Phone: 8248227615
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 10_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : COD

1. Problem Statement

Priya is analyzing encrypted messages in a research project. She wants to analyze the frequency of each character in a given paragraph. The characters should be stored in a TreeMap so that the output is sorted in ascending order of characters automatically.

You are required to build a Java program that:

Uses a TreeMap<Character, Integer> to count how many times each character appears in the message.Ignores spaces and considers only alphabets (case-sensitive).Outputs the frequencies of characters in sorted order.

You must use a TreeMap in the class named MessageAnalyzer.

Input Format

The first line of input contains an integer n, the number of lines in the message.

The next n lines each contain a string (the encrypted message line).

Output Format

The first line of output prints: "Character Frequency:"

Then print each character and its frequency in the format: "<character>: <count>"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
Hello World
Java
Output: Character Frequency:
H: 1
J: 1
W: 1
a: 2
d: 1
e: 1
l: 3
o: 2
r: 1
v: 1

Answer

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

class MessageAnalyzer {
    public static void analyzeMessage(int n, Scanner sc) {
        TreeMap<Character, Integer> freqMap = new TreeMap<>();

        for (int i = 0; i < n; i++) {
            String line = sc.nextLine();
            for (char c : line.toCharArray()) {
                if (Character.isAlphabetic(c)) {
```

```
                freqMap.put(c, freqMap.getOrDefault(c, 0) + 1);
            }
        }
    }

    System.out.println("Character Frequency:");
    for (Map.Entry<Character, Integer> entry : freqMap.entrySet()) {
        System.out.println(entry.getKey() + ": " + entry.getValue());
    }
  }
}

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    sc.nextLine(); // consume newline
    MessageAnalyzer.analyzeMessage(n, sc);
    sc.close();
  }
}
```

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

Rajalakshmi Engineering College

Name: Rahul G
Email: 241901087@rajalakshmi.edu.in
Roll no: 241901087
Phone: 8248227615
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 10_Q4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : COD

1. Problem Statement

In a ticket reservation system, you store the available seat numbers in a TreeSet. Users input their desired seat number, and the program checks whether the chosen seat is available.

Using a TreeSet ensures quick and efficient verification of seat availability, ensuring a smooth and organized ticket booking process.

Input Format

The first line of input contains a single integer n, representing the number of available seats.

The second line contains n space-separated integers, representing the available seat numbers.

The third line contains an integer m, representing the seat number that needs to be searched.

Output Format

The output displays "[m] is present!" if the given seat is available. Otherwise, it displays "[m] is not present!"

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
2 4 5 6
5
Output: 5 is present!

Answer

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

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

        for (int i = 0; i < n; i++) {
            seats.add(sc.nextInt());
        }

        int m = sc.nextInt();

        if (seats.contains(m)) {
            System.out.println(m + " is present!");
        } else {
            System.out.println(m + " is not present!");
        }

        sc.close();
    }
```

}

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Rahul G
Email: 241901087@rajalakshmi.edu.in
Roll no: 241901087
Phone: 8248227615
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 10_PAH

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Riya is building a calendar event scheduler where each event is stored in chronological order using a TreeMap. The key represents the event time in 24-hour format (HH:MM), and the value is the event description.

She wants the system to:

Automatically sort events by time.Avoid duplicate time entries — if a duplicate time is entered, ignore the new entry.Print all scheduled events in order.

Implement this logic using a class named EventManager.

Input Format

The first line of the input contains an integer n, representing the number of events.

The next n lines each contain a string in the format: "HH:MM Description"

(Example: 09:00 TeamMeeting).

Output Format

The first line of the output prints "Scheduled Events:"

The next k lines print each event in the format: "HH:MM - Description"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
09:00 TeamMeeting
13:30 LunchBreak
11:00 ProjectUpdate
09:00 Standup
15:00 ClientCall

Output: Scheduled Events:
09:00 - TeamMeeting
11:00 - ProjectUpdate
13:30 - LunchBreak
15:00 - ClientCall

Answer

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

class EventManager {
    public static void scheduleEvents(int n, Scanner sc) {
        TreeMap<String, String> events = new TreeMap<>();

        for (int i = 0; i < n; i++) {
            String time = sc.next();
            String description = sc.next();
            if (!events.containsKey(time)) {
                events.put(time, description);
            }
```

```
        }

        System.out.println("Scheduled Events:");
        for (Map.Entry<String, String> entry : events.entrySet()) {
            System.out.println(entry.getKey() + " - " + entry.getValue());
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        EventManager.scheduleEvents(n, sc);
        sc.close();
    }
}
```

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

2. Problem Statement

A university maintains a list of student records and wants to store them in a sorted manner based on their GPA. If two students have the same GPA, they should be further sorted by their name in lexicographical order. Implement a program that uses a TreeSet to store student records and ensures unique student IDs.

Input Format

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

The next N lines contain details of each student in the format: "StudentID Name GPA"

- StudentID (Integer) - A unique identifier.
- Name (String) - The student's name (can contain spaces).
- GPA (Double) - The Grade Point Average.

Output Format

The output prints the list of students in ascending order of GPA.

If two students have the same GPA, sort them by name.

Print details in the format: "StudentID Name GPA" in the output, GPA is rounded to two decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
101 John 8.5
102 Alice 9.1
103 Bob 8.5
104 Zoe 7.3
105 Charlie 9.1

Output: 104 Zoe 7.30
103 Bob 8.50
101 John 8.50
102 Alice 9.10
105 Charlie 9.10

Answer

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

class Student implements Comparable<Student> {
    int id;
    String name;
    double gpa;

    Student(int id, String name, double gpa) {
        this.id = id;
        this.name = name;
        this.gpa = gpa;
    }

    @Override
    public int compareTo(Student s) {
        if (Double.compare(this.gpa, s.gpa) != 0)
```

```
            return Double.compare(this.gpa, s.gpa);
        else
            return this.name.compareTo(s.name);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Student)) return false;
        Student s = (Student) o;
        return this.id == s.id;
    }

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

    @Override
    public String toString() {
        return id + " " + name + " " + String.format("%.2f", gpa);
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine(); // consume newline

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

        for (int i = 0; i < n; i++) {
            String line = sc.nextLine().trim();
            String[] parts = line.split(" ");
            int id = Integer.parseInt(parts[0]);
            double gpa = Double.parseDouble(parts[parts.length - 1]);

            // Extract name (which can have spaces)
            StringBuilder nameBuilder = new StringBuilder();
            for (int j = 1; j < parts.length - 1; j++) {
                nameBuilder.append(parts[j]);
```

```
            if (j != parts.length - 2)
                nameBuilder.append(" ");
            }
            String name = nameBuilder.toString();

            students.add(new Student(id, name, gpa));
        }

        for (Student s : students) {
            System.out.println(s);
        }

        sc.close();
    }
}
```

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

3. Problem Statement

Sarah is working on a spam detection system that analyzes incoming messages for unique patterns. Spammers often use repetitive character sequences, making it important to identify the first non-repeating character in a message.

Given a string, Sarah needs to determine the first character that appears only once. If all characters repeat, the system should return -1.

She decides to use a HashMap to efficiently track character frequencies and find the solution.

Input Format

The first line contains an integer N representing , the length of the string.

The second line contains a string of N lowercase English letters (a-z).

Output Format

The output prints a character representing the first non-repeating character. If none exist, print -1.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10
abacabadac

Output: d

Answer

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

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        String str = sc.next();

        HashMap<Character, Integer> freq = new HashMap<>();

        // Count frequency of each character
        for (char c : str.toCharArray()) {
            freq.put(c, freq.getOrDefault(c, 0) + 1);
        }

        // Find the first non-repeating character
        char result = '-';
        for (char c : str.toCharArray()) {
            if (freq.get(c) == 1) {
                result = c;
                break;
            }
        }

        if (result == '-') {
            System.out.println(-1);
        } else {
            System.out.println(result);
        }
```

```
        sc.close();
    }
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Rahul G
Email: 241901087@rajalakshmi.edu.in
Roll no: 241901087
Phone: 8248227615
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

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 : 40

Section 1 : COD

1. 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

```
import java.util.*;
```

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

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

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Book)) return false;
        Book b = (Book) o;
        return this.isbn == b.isbn;
    }

    @Override
    public int hashCode() {
        return Objects.hash(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) {
        books.add(new Book(isbn, title, author));
    }

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

    public void displayBooks() {
```

```
        if (books.isEmpty()) {
            System.out.println("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**

2. 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;

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

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

    @Override
    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;
    }

    @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<>();
        HashMap<Integer, Student> studentMap = new HashMap<>();

        for (int i = 0; i < n; i++) {
            String cmd = sc.next();

            if (cmd.equals("A")) {
                int roll = sc.nextInt();
                String name = sc.next();
                if (!studentMap.containsKey(roll)) {
                    Student s = new Student(roll, name);
                    students.add(s);
                    studentMap.put(roll, s);
                }
            }
            else if (cmd.equals("M")) {
                int roll = sc.nextInt();
                if (studentMap.containsKey(roll)) {
                    studentMap.get(roll).attendance++;
                }
            }
            else if (cmd.equals("D")) {
                for (Student s : students) {
                    System.out.println(s);
                }
            }
        }

        sc.close();
    }
}
```

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

3. 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 {
            double avg = (double) totalSum / totalCount;
            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**

4. Problem Statement

A linguist named Meera is classifying a list of words based on their first character. She wants to store words grouped by their starting letter using a

TreeMap so that the groups appear in sorted order of characters (i.e., 'a' to 'z'). For each letter, all words starting with that letter should be stored in the order they appear.

Implement the logic inside a class named WordClassifier using the TreeMap<Character, List<String>> collection.

Input Format

The first line of the input contains an integer n, representing the number of words.

The next n lines each contain a word.

Output Format

The first line of the output prints: "Grouped Words by Starting Letter:"

The next lines print each character key and its list of words in the format:

"letter: word1 word2 word3...

..."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
dog
deer
cat
cow
camel

Output: Grouped Words by Starting Letter:
c: cat cow camel
d: dog deer

Answer

```
import java.util.*;
```

```
class WordClassifier {
    public void classifyWords(List<String> words) {
        TreeMap<Character, List<String>> map = new TreeMap<>();

        for (String word : words) {
            char firstChar = word.charAt(0);
            map.putIfAbsent(firstChar, new ArrayList<>());
            map.get(firstChar).add(word);
        }

        System.out.println("Grouped Words by Starting Letter:");
        for (Map.Entry<Character, List<String>> entry : map.entrySet()) {
            System.out.print(entry.getKey() + ": ");
            for (String w : entry.getValue()) {
                System.out.print(w + " ");
            }
            System.out.println();
        }
    }
}

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

        List<String> words = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            words.add(sc.nextLine());
        }

        WordClassifier classifier = new WordClassifier();
        classifier.classifyWords(words);
    }
}
```

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