

# Rajalakshmi Engineering College

Name: Prince Rohith  
Email: 240701399@rajalakshmi.edu.in  
Roll no: 240701399  
Phone: 6369941431  
Branch: REC  
Department: CSE - Section 5  
Batch: 2028  
Degree: B.E - CSE

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**

```
import java.util.Scanner;  
import java.util.TreeMap;
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        int n = scanner.nextInt();  
        scanner.nextLine();  
        TreeMap<String, String> events = new TreeMap<>();  
        for (int i = 0; i < n; i++) {  
            String line = scanner.nextLine().trim();  
            String[] parts = line.split(" ", 2);
```

```

        if (parts.length == 2) {
            String time = parts[0];
            String description = parts[1];

            if (!events.containsKey(time)) {
                events.put(time, description);
            }
        }
    }

    System.out.println("Scheduled Events:");
    for (String time : events.keySet()) {
        System.out.println(time + " - " + events.get(time));
    }

    scanner.close();
}

```

**Status :** Correct

**Marks : 10/10**

## 2. 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**

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

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

        if (str.length() != N) {
            System.out.println(-1);
            scanner.close();
            return;
        }

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

        for (char c : str.toCharArray()) {
            freq.put(c, freq.getOrDefault(c, 0) + 1);
        }

        char result = '\0';
        for (char c : str.toCharArray()) {
            if (freq.get(c) == 1) {
                result = c;
            }
        }

        System.out.println(result);
    }
}
```

```

        break;
    }
}

if (result == '\0') {
    System.out.println(-1);
} else {
    System.out.println(result);
}

scanner.close();
}
}

```

**Status :** Correct

**Marks :** 10/10

### 3. 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**

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

```
import java.util.TreeSet;
```

```
class Student implements Comparable<Student> {
```

```
    int id;
```

```
    String name;
```

```
    double gpa;
```

```
    public Student(int id, String name, double gpa) {
```

```
        this.id = id;
```

```
        this.name = name;
```

```
        this.gpa = gpa;
```

```
    }
```

```
    @Override
```

```
    public int compareTo(Student other) {
```

```
        if (this.gpa != other.gpa) {
```

```
            return Double.compare(this.gpa, other.gpa);
```

```
        }
```

```
        return this.name.compareTo(other.name);
```

```
    }
```

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

```
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int N = Integer.parseInt(scanner.nextLine().trim());
        TreeSet<Student> students = new TreeSet<>();

        for (int i = 0; i < N; i++) {
            String line = scanner.nextLine().trim();
            String[] parts = line.split(" ", 3);

            int id = Integer.parseInt(parts[0]);
            String name = parts[1];
            double gpa = Double.parseDouble(parts[2]);

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

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

        scanner.close();
    }
}
```

**Status :** Correct

**Marks :** 10/10