# Rajalakshmi Engineering College

Name: Samyuktha S
Email: 240701701@rajalakshmi.edu.in
Roll no: 240701701
Phone: 6380226314
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 : 40

## 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.*;
import java.text.DecimalFormat;
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);
         }
         System.out.println();
      } else {
         // Recalculate sum and count if needed (or use passed sum and totalCount)
         long combinedSum = sum; // you can use the passed sum if it already includes both sets
```

```
            int combinedCount = totalCount; // or calculate size of both sets
            double avg = (double) combinedSum / combinedCount;
            DecimalFormat df = new DecimalFormat("0.00");
            System.out.println("NO " + df.format(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

Tony is an e-learning platform administrator, he oversees the user ratings for various online courses offered in the platform.

To enhance user experience, you should assist him in utilizing a HashMap to store course ratings given by learners. Regularly, he analyzes this data to identify the highest and lowest-rated courses, enabling targeted

improvements and ensuring the quality of the educational content. This process assists in maintaining a competitive and engaging online learning environment for the users.

***Input Format***

The input consists of a string representing the course name followed by a double value representing the course's rating, in separate lines.

The input is terminated by entering "done".

***Output Format***

The first line of output prints the string "Highest Rated Course: " followed by the highest-rated course.

The second line prints the string "Lowest Rated Course: " followed by the lowest-rated courses.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: DSA
4.0
OOPS
4.2
C
3.2
done

Output: Highest Rated Course: OOPS
Lowest Rated Course: C

***Answer***

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

class CourseAnalyzer {
```

```
    public Map<String, String>
identifyHighestAndLowestRatedCourses(Map<String, Double> courseRatings) {
        Map<String, String> result = new HashMap<>();
        if (courseRatings.isEmpty()) {
            result.put("highest", "");
            result.put("lowest", "");
            return result;
        }

        double maxRating = Double.NEGATIVE_INFINITY;
        double minRating = Double.POSITIVE_INFINITY;
        String highestCourse = "";
        String lowestCourse = "";

        for (Map.Entry<String, Double> entry : courseRatings.entrySet()) {
            String course = entry.getKey();
            double rating = entry.getValue();

            if (rating > maxRating) {
                maxRating = rating;
                highestCourse = course;
            }

            if (rating < minRating) {
                minRating = rating;
                lowestCourse = course;
            }
        }

        result.put("highest", highestCourse);
        result.put("lowest", lowestCourse);

        return result;
    }
}


public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Map<String, Double> courseRatings = new HashMap<>();

        while (true) {
```

```
            String courseName = scanner.nextLine();
            if (courseName.equalsIgnoreCase("done")) {
                break;
            }
            double rating = Double.parseDouble(scanner.nextLine().trim());
            courseRatings.put(courseName, rating);
        }

        CourseAnalyzer analyzer = new CourseAnalyzer();
        Map<String, String> result =
analyzer.identifyHighestAndLowestRatedCourses(courseRatings);

        System.out.printf("Highest Rated Course: %s\n", result.get("highest"));
        System.out.printf("Lowest Rated Course: %s", result.get("lowest"));

        scanner.close();
    }
}
```

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

3. Problem Statement

Bob wants to develop a score-tracking application for a gaming tournament. Each player's score is stored in a HashMap with the player's name as the key and the score as the value.

Write a program to assist Bob that takes user input to enter player scores, calculates the maximum score from the HashMap, and prints the player with the highest score.

***Input Format***

The input consists of strings representing player details in the format "playerName:score".

The input is terminated by entering "done".

***Output Format***

The output displays a string, representing the player's name who scored the maximum.

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

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

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: Alice:15
Bob:56
done
Output: Bob

***Answer***

```
import java.util.*;

class ScoreTracker {
  HashMap<String, Integer> scoreMap = new HashMap<>();


   public boolean processInput(String input) {

      if (!input.matches("^[A-Za-z]+:[0-9]+$")) {

         if (!input.contains(":")) {
            System.out.println("Invalid format");
         } else {
            String[] parts = input.split(":");
            if (parts.length != 2 || !parts[0].matches("^[A-Za-z]+$")) {
               System.out.println("Invalid format");
            } else if (!parts[1].matches("[0-9]+")) {
               System.out.println("Invalid input");
            } else {
               System.out.println("Invalid format");
            }
         }
         return false;
      }
```

```
        String[] parts = input.split(":");
        String name = parts[0];
        String scoreStr = parts[1];

        try {
            int score = Integer.parseInt(scoreStr);

            if (score < 1 || score > 100 || name.length() > 20) {
                System.out.println("Invalid input");
                return false;
            }
            scoreMap.put(name, score);
        } catch (NumberFormatException e) {
            System.out.println("Invalid input");
            return false;
        }
        return true;
    }


    public String findTopPlayer() {
        String topPlayer = "";
        int maxScore = Integer.MIN_VALUE;
        for (Map.Entry<String, Integer> entry : scoreMap.entrySet()) {
            if (entry.getValue() > maxScore) {
                maxScore = entry.getValue();
                topPlayer = entry.getKey();
            }
        }
        return topPlayer;
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        ScoreTracker tracker = new ScoreTracker();
        boolean validInput = true;

        while (true) {
            String input = scanner.nextLine();
```

```
            if (input.toLowerCase().equals("done")) {
                break;
            }

            if (!tracker.processInput(input)) {
                validInput = false;
                break;
            }
        }

        if (validInput && !tracker.scoreMap.isEmpty()) {
            System.out.println(tracker.findTopPlayer());
        }

        scanner.close();
    }
}
```

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

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***

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


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

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++) {
            String command = sc.next();

            if (command.equals("A")) {
                int rollNo = sc.nextInt();
                String name = sc.next();


                boolean exists = false;
                for (Student s : students) {
                    if (s.rollNo == rollNo) {
                        exists = true;
                        break;
                    }
                }

                if (!exists) {
                    students.add(new Student(rollNo, name));
                }
```

```
        } else if (command.equals("M")) {
            int rollNo = sc.nextInt();
            for (Student s : students) {
                if (s.rollNo == rollNo) {
                    s.attendance++;
                    break;
                }
            }

        } else if (command.equals("D")) {
            for (Student s : students) {
                System.out.println(s.rollNo + " " + s.name + " " + s.attendance);
            }
        }
    }

    sc.close();
  }
}
```

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