QUESTION 1: Library Book Borrowing Tracker Using HashMap

You are building a system to track books borrowed from a library. The system should store the titles of books along with the number of times they have been borrowed. Users should be able to add, update, or remove book records, check if a book exists, and retrieve specific borrow counts. Additionally, the system should calculate the total number of books borrowed from the library.

If any operation attempts to access a book that does not exist in the system, a custom exception, **BookNotFoundException**, should be thrown and handled appropriately.

Operations to be Performed

1. **Add Book Records**: Input the number of books and their respective titles and borrow counts.

2. **Retrieve Borrow Count**: Look up the total number of times a specific book has been borrowed by its title.

3. **Remove Book Records**: Remove a book's record from the system.

4. **Check if a Book Exists**: Verify if a specific book is in the system based on its title.

5. **Calculate Total Borrow Count**: Display the total number of times books have been borrowed.

Input Format

Adding Books:

• The first line contains an integer n, representing the number of books to add.

• For each book, input:

  o The first line contains the book's title (String).

  o The second line contains the borrow count (Integer).

Perform Operations:

• A line containing the title of a book to retrieve its borrow count.

• A line containing the title of a book to remove from the system.

• A line containing the title of a book to check if it exists in the system.

Output Format

Retrieving Borrow Count:

• If found, print: [Book title] has been borrowed [count] times.

• If not found, throw and catch the BookNotFoundException and print: Book [Book title] not found.

Removing a Book:

• If found and removed, print: Book [Book title] has been removed from the library.

• If not found, throw and catch the BookNotFoundException and print: Book [Book title] not found for removal.

Checking if a Book Exists:

• If the book exists, print: Book [Book title] exists in the library.

• If the book doesn’t exist, throw and catch the BookNotFoundException and print: Book [Book title] does not exist in the library.

Calculating Total Borrow Count:

• Print the total borrow count of all books: Total Books Borrowed: [total count]

Sample Input 1

3

The Alchemist

150

To Kill a Mockingbird

200

1984

100

The Alchemist To Kill a Mockingbird

1984

Sample Output 1

The Alchemist has been borrowed 150 times.

Book To Kill a Mockingbird has been removed from the library.

Book 1984 exists in the library.

Total Books Borrowed: 250

Sample Input 2

2

The Great Gatsby

250

Pride and Prejudice

300

Moby Dick

War and Peace

Anna Karenina

Sample Output 2

Book Moby Dick not found.

Book War and Peace not found for removal.

Book Anna Karenina does not exist in the library.

Total Books Borrowed: 550

QUESTION 2: Employee Record Management System using HashMap

You are developing an Employee Record Management System for a corporate organization. The system should manage employee records using their ID and name. Users can add new employees, search for a specific employee, display all current employee records, and get the total number of active employees. After all operations, the system should also identify and display the employee with the longest name.

Input Format:

- The first line contains an integer `n`, representing the number of employee records.
- The next `2 * n` lines provide the details of each employee:
- A line containing the employee ID (String).
- A line containing the employee name (String).
- The last line of input contains a String representing the employee ID to search for.

Output Format:

Employee Search Result:

o If the employee ID is found, print: `Employee with ID [employee ID] is named: [employee name]`

o If the employee ID is not found, print: `No employee found with ID: [employee ID]`

Total Number of Employees:

o Print: `Total number of employees: [number]`

Display All Employee Records:

o List all employees in the format: `Employee ID: [employee ID], Name: [employee name]`

Employee with Longest Name:

o Print: `Employee with the longest name is: [employee name]`

Sample Input:

3

E101

John Doe

E202

Jane Smith

E303

Alexander Hamilton

E202

Sample Output:

Employee with ID E202 is named: Jane Smith

Total number of employees: 3

Employee Records:

Employee ID: E101, Name: John Doe

Employee ID: E202, Name: Jane Smith

Employee ID: E303, Name: Alexander Hamilton

Employee with the longest name is: Alexander Hamilton

QUESTION 3: Library Book Catalogue Management System using HashMap

You are developing a Library Book Catalogue Management System for a library. The system should manage books using their ID and title. Users can add new books, search for a specific book, display all current books, and get the total number of books in the catalogue. After all operations, the system should also identify and display the book with the longest title.

Input Format:

- The first line contains an integer `n`, representing the number of books in the catalog.
- The next `2 * n` lines provide the details of each book:
  - A line containing the book ID (String).
  - A line containing the book title (String).
- The last line of input contains a String representing the book ID to search for.

Output Format:

Book Search Result:

o If the book ID is found, print: `Book with ID [book ID] is tled: [book tle]`

o If the book ID is not found, print: `No book found with ID: [book ID]`

Total Number of Books:

o Print: `Total number of books: [number]`

Display All Books:

o List all books in the format: `Book ID: [book ID], Title: [book tle]`

Book with Longest Title:

o Print: `Book with the longest title is: [book title]`

Sample Input:

4

B101

The Great Gatsby

B102

To Kill a Mockingbird

B103

1984

B104

A Brief History of Time

B102

Sample Output:

Book with ID B102 is tled: To Kill a Mockingbird

Total number of books: 4

Book Catalog:

Book ID: B101, Title: The Great Gatsby

Book ID: B102, Title: To Kill a Mockingbird

Book ID: B103, Title: 1984

Book ID: B104, Title: A Brief History of Time

Book with the longest title is: A Brief History of Time

QUESTION 4: Shopping Cart Discount Management using HashMap

You are tasked with developing a simple shopping cart management system for a retail store. The system should allow the user to input the names and prices of items they want to purchase. After the data is collected, the system should ask for a total price threshold. If the total price of items purchased exceeds the threshold, the customer will receive a 10% discount on the total bill. The program will ensure that only positive price values are accepted and will display the details of the purchased items, the total bill before discount, and the total bill after applying the discount if applicable.

Input Format

- The first line of input consists of an integer representing the number of items the customer wants to purchase.
- The next two lines consist of the user inputs for each item:
    - Item name (String)
    - Item price (double, must be a positive number)
- After entering the item details, the last line of input consists of the total price threshold.

Output Format

- The output displays the names and prices of the items (2 decimal values) purchased.
- The total bill (2 decimal values) before any discount.
- If the total bill exceeds the threshold, the output displays the total bill (2 decimal values) after a 10% discount. if not then "No discount applied as the total bill does not exceed the threshold."
- If no items are purchased, the output prints a message stating that "No items were purchased".

Sample Input

2

Banana

0.99

Juice

3.50

10.00

Sample Output

Items purchased:

Item: Juice, Price: 3.50

Item: Banana, Price: 0.99

Total bill before discount: 4.49

No discount applied as the total bill does not exceed the threshold.

QUESTION 5: Employee Salary Management using HashMap

You are tasked with developing a simple employee salary management system for a company. The system should allow the HR manager to input the names and salaries of employees. After the data is collected, the system should ask for a salary threshold and then display the names of all employees whose salaries are above that threshold. The program will ensure that only positive salary values are accepted and will display a message if no employees meet the salary criteria.

Input Format

- The first line of input consists of an integer representing the number of employees.
- The next two lines consist of the user inputs for each employee:
    - Employee name (String)
    - Employee salary (double, must be a positive number)
- After entering employee details, the last line of input consists of the salary threshold.

Output Format

- The output displays the names and salaries of employees whose salaries are greater than the provided threshold.
- Please note that only upto two decimal places are allowed while printing the salary.
- If no employees meet the criteria, the output prints a message stating that "No employees have a salary greater than the threshold".

Sample Input

1

Ajay

23500.50

25500.00

Sample Output

Employee: Pranab, Salary: 32500.23

QUESTION 6: WORD COUNTER

Imagine you are developing a text processing application where you need to analyse the frequency of each character in a user- provided text input. Your application aims to provide statistical insights into the textual content, such as character frequency distributions, without distinguishing between uppercase and lowercase letters using HashMap

You need to implement a Java method that counts the occurrences of each character in a given string, treating uppercase and lowercase versions of the same character as identical.

Input Format

It will take user input as string

Output Format

It should print each char and the count of the particular

QUESTION 7: Caregiver Shift Management System Using HashMap

You are developing a system to manage caregiver duty schedules at an assisted living facility. The system should allow users to input the names of caregivers and the number of days they have worked during a month. This information should be stored in a HashMap. The system must perform the following operations to manage and query caregiver duty records:

Operations to be performed:

1. **Add Caregiver Records**: Input the number of caregivers and their respective names along with the number of days worked.
2. **Retrieve Caregiver's Working Days**: Look up how many days a specific caregiver has worked by their name.
3. **Remove Caregiver Records**: Remove a caregiver's record from the schedule.
4. **Check if a Caregiver Exists**: Verify if a specific caregiver is in the system based on their name.
5. **Count Total Days Worked**: Calculate and display the total number of days worked by all caregivers.

Input Format:

1. Adding Caregivers to the System:
    - o The first line contains an integer n representing the number of caregivers on duty (between 1 and 12).
    - o For each caregiver, input two lines:
        - ▪ The first line contains the caregiver's name (String).
        - ▪ The second line contains the number of days worked (an integer between 1 and 30).
2. Perform Operations:
    - o A line containing the name of a caregiver to look up how many days they worked.
    - o A line containing the name of a caregiver to remove from the system.
    - o A line containing the name of a caregiver to check if they exist in the system.

Output Format:

1. For retrieving the number of days a caregiver worked:
    - o If the caregiver is found, print: [Caregiver name] worked for [days worked] days.
    - o If the caregiver is not found, print: Caregiver [Caregiver name] not found.
2. For removing a caregiver:
    - o If the caregiver is found and removed, print: Caregiver [Caregiver name] has been removed from the schedule.
    - o If the caregiver is not found, print: Caregiver [Caregiver name] not found for removal.
3. For checking if a caregiver exists:
    - o If the caregiver exists, print: Caregiver [Caregiver name] exists in the system.
    - o If the caregiver does not exist, print: Caregiver [Caregiver name] does not exist in the system.
4. For counting total days worked:

o Print the total number of days worked by all caregivers; Total Days Worked: [total days worked].

Error Handling:

Ensure that the number of days worked is between 1 and 30. If invalid, output: Days worked must be between 1 and 30.

Sample 1 Input:

3

Laura

12

John

20

Megan

10

John

Megan

Laura

Sample 1 Output:

John worked for 20 days.

Caregiver Megan has been removed from the schedule.

Caregiver Laura exists in the system.

Total Days Worked: 32

Sample 2 Input:

2

Elena

25

Noah

18

Sophia

Liam

Isabella

Sample 2 Output:

Caregiver Sophia not found.

Caregiver Liam not found for removal.

Caregiver Isabella does not exist in the system.

Total Days Worked: 43

Sample 3 Input:

3

Michael

10

Emma

50

Olivia

7

Lucas

Olivia

Lucas

Sample 3 Output:

Days worked must be between 1 and 30.

Caregiver Olivia has been removed from the schedule.

Caregiver Lucas not found.

Total Days Worked: 10

QUESTION 8: Farmland Harvest Tracking Using HashMap

You are developing a system to manage and track harvest yields for different plots on a farm. The system should allow users to input plot names and their respective harvest yields (in tons) and store this information using a HashMap. The system should also allow users to query and manage the harvest yield records by performing the following operations:

Operations to be Performed:

1. **Add Plot Harvest Records:**
    - o Input plot names and their harvest yields (in tons) and store them in the system.
2. **Retrieve Harvest Yield by Plot:**
    - o Look up and display the harvest yield for a specific plot using its name.
3. **Check if a Certain Harvest Yield Exists:**
    - o Verify if any plot has a harvest yield of a specified value (in tons).
4. **Check the Total Number of Plots:**
    - o Calculate and display the total number of plots stored in the system in ascending order of harvest yields (in tons).
5. **Display All Plot Names and Yields:**
    - o Output all the plots with their corresponding harvest yields.

Input Format:

1. Adding Plot Harvest Records:
    - o The first line contains an integer n representing the number of plots to be recorded (between 1 and 20).
    - o For each plot, input two lines:
        - ▪ The first line contains the plot name (String).
        - ▪ The second line contains the harvest yield in tons (a positive number).
2. Performing Operations:
    - o A line containing the name of a plot to retrieve its harvest yield.
    - o A line containing a harvest yield value (in tons) to check if any plot has this yield.

Output Format:

1. For retrieving harvest yield by plot:
    - o If the plot is found, print: [Plot name] has a harvest yield of [yield] tons.
    - o If the plot is not found, print: Plot [Plot name] not found.
2. For checking if a certain harvest yield exists:
    - o If the harvest yield exists, print: A plot with a harvest yield of [yield] tons exists.

o If the harvest yield does not exist, print: No plot has a harvest yield of [yield] tons.

3. For getting the total number of plots:

o Print the total number of plots and display them in the following format: Total number of plots: [number of plots].

4. For displaying all plot names and yields:

o Output each plot name and its corresponding harvest yield in ascending order as follows: Plot: [Plot name], Yield: [yield] tons

---

Error Handling:

• If the harvest yield is a negative value or zero, output: "Harvest yield must be a positive number." and stop the process. Refer to the sample output for formatting specifications.

Sample 1 Input:

2

NorthPlot

350

SouthPlot

400

NorthPlot

400

Sample 1 Output:

NorthPlot has a harvest yield of 350 tons.

A plot with a harvest yield of 400 tons exists.

Total number of plots: 2

Plot: NorthPlot, Yield: 350 tons

Plot: SouthPlot, Yield: 400 tons

Sample 2 Input:

2

PlotX

120

PlotY

180

PlotZ

250

Sample 2 Output:

Plot PlotZ not found.

No plot has a harvest yield of 250 tons.

Total number of plots: 2

Plot: PlotX, Yield: 120 tons

Plot: PlotY, Yield: 180 tons

QUESTION 9: Environmental Sustainability Program Management System

You are developing an Environmental Sustainability Program Management System for an organization focused on promoting sustainability initiatives. The system should manage different sustainability programs for various regions. Users can add new programs, search for a specific program, display all current programs, and get the total number of active programs. After all operations, the system should also identify and display the program with the longest name.

Input Format:

1. The first line contains an integer n, representing the number of sustainability programs.

2. The next 2 * n lines provide the details of each program:

    o A line containing the program ID (String).

    o A line containing the program name (String).

3. The last line of input contains a String representing the program ID to search for.

Output Format:

Program Search Result:

- If the program ID is found, print: Program with ID [program ID] is named: [program name]
- If the program ID is not found, print: No program found with ID: [program ID]

Total Number of Programs:

- Print: Total number of programs: [number]

Display All Programs:

- Display all programs with the heading: Sustainability programs: followed by each program in the format:

  Program ID: [program ID], Name: [program name]

Program with the Longest Name:

- Print: Program with the longest name is: [program name]

Sample 1 Input:

3

S101

Green Energy Initiative

S202

Zero Waste Campaign

S303

Clean Oceans Movement

S202

Sample 1 Output:

Program with ID S202 is named: Zero Waste Campaign

Total number of programs: 3

Sustainability programs:

Program ID: S202, Name: Zero Waste Campaign

Program ID: S303, Name: Clean Oceans Movement

Program ID: S101, Name: Green Energy Initiative

Program with the longest name is: Clean Oceans Movement

Sample 2 Input:

3

S101

Protect Our Wildlife

S202

Plastic-Free Future

S303

Global Climate Action Plan

S404

Sample 2 Output:

No program found with ID: S404

Total number of programs: 3

Sustainability programs:

Program ID: S202, Name: Plastic-Free Future

Program ID: S303, Name: Global Climate Action Plan

Program ID: S101, Name: Protect Our Wildlife

Program with the longest name is: Global Climate Action Plan

Question 10: Staff Duty Management System Using HashMap

You are tasked with creating a system to manage duty schedules for staff in a facility. This system should enable users to enter the names of staff members along with the number of days they have worked in a month. The information should be stored in a HashMap. The system must support the following operations for managing and querying staff duty records:

Operations to Perform:

1. **Add Staff Records**: Input the total number of staff members and their corresponding names along with the number of days worked.

2. **Retrieve Staff's Working Days**: Query how many days a specific staff member has worked using their name.

3. **Remove Staff Records**: Delete a staff member's record from the schedule.

4. **Check Staff Existence**: Confirm if a specific staff member is present in the system using their name.

5. **Count Total Days Worked**: Calculate and display the total number of days worked by all staff members.

Input Format

Adding Staff to the System:

• The first line contains an integer n indicating the number of staff members in the shift.

For each staff member, provide two lines of input:

• The first line contains the staff member's name (String).

• The second line contains the number of days worked.

Perform Operations:

• A line containing the name of a staff member to check how many days they have

worked.

• A line containing the name of a staff member to remove from the system.

• A line containing the name of a staff member to verify if they exist in the system.

Output Format

• For retrieving the number of days a staff member worked:

o If the staff member is found, print: [Staff member name] worked for [days

worked] days.

o If the staff member is not found, print: Staff member [Staff member

name] not found.

• For removing a staff member:

o If the staff member is found and removed, print: Staff member [Staff

member name] has been removed from the schedule.

o If the staff member is not found, print: Staff member [Staff member

name] not found for removal.

• For checking if a staff member exists:

o If the staff member exists, print: Staff member [Staff member name] exists in the system.

o If the staff member does not exist, print: Staff member [Staff member name] does not exist in the system.

• For counting total days worked:

o Print the total number of days worked by all staff members: Total Days Worked: [total days worked]

Sample Input 1

3

Alice

10

Bob

8

Carol

15

Bob

Carol

Alice

Sample Output 1

Bob worked for 8 days.

Staff member Carol has been removed from the schedule.

Staff member Alice exists in the system.

Total Days Worked: 18

Sample Input 2

2

Eve

25

Zara

20

Mia

Ziva

Ava

Sample Output 2

Staff member Mia not found.

Staff member Ziva not found for removal.

Staff member Ava does not exist in the system.

Total Days Worked: 45

Sample Input 3

3

Alice

10

Bob

41

Catherine

5

David

Catherine

David

Sample Output 3

Staff member David not found.

Staff member Catherine has been removed from the schedule.

Staff member David does not exist in the system.

Total Days Worked: 51