Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 1_Q1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Gloria is responsible for monitoring the performance of two machines in a factory. She needs to determine which of the two machines is operating closest to the optimal temperature of 100 degrees Celsius using the relational operator.

Assist Gloria in displaying the machine's temperature, which is closer to 100, and the difference from 100.

Input Format

The first line of input consists of an integer N, representing the temperature of the first machine.

The second line consists of an integer M, representing the temperature of the second machine.

Output Format

The output prints "The integer closer to 100 is X with a difference of Y" where X is the temperature of the closer machine and Y is the difference from 100.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 90
80

Output: The integer closer to 100 is 90 with a difference of 10

Answer

```
import java.io.*;
import java.util.*;
public class Main {
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int X, Y;
      int N = sc.nextInt();
      int M = sc.nextInt();
      int difference1 = 100-N;
      int A = Math.abs(difference1);
      int difference2 = 100-M;
      int B = Math.abs(difference2);
      if(A <= B){
         X = N;}
      else{
         X = M;}
      Y = Math.abs(100-X);
      System.out.println("The integer closer to 100 is "+X+" with a difference of "+Y
+"");

   }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 1_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. PROBLEM STATEMENT:

Dave got two students who wants help with their doubt. Each handouts an integer and wants to find if one Integer Positive While the Other is Not Divisible by 3. Write a program to achieve this and conclude for them.

Input Format

The first line of input represents the first integer.

The second line of input represents the second integer.

Output Format

The output should display as "One of the integers is positive while the other is not divisible by 3." or "Neither of the integers meets the condition."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
3
Output: One of the integers is positive while the other is not divisible by 3.

Answer

```
import java.io.*;
import java.util.*;
public class Main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int N = sc.nextInt();
      int M = sc.nextInt();
      if ((N>0 && M%3!=0) || (M>0 && N%3!=0)) {
         System.out.println("One of the integers is positive while the other is not
divisible by 3.");
      }
      else {
        System.out.println("Neither of the integers meets the condition.");
      }
   }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 1_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem statement

Manoj, a developer at MoneyMatters Inc., is working on improving the company's financial system. He needs to create a program that takes an integer input, converts it into a double, and displays both the original integer and the converted double value.

Input Format

The input consists of a single integer representing a monetary amount.

Output Format

The first line of the output displays the "Original Integer: ", followed by an integer representation of the input value.

The second line displays the "Converted Double: ", followed by a double value representing the input as a decimal value.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 20

Output: Original Integer: 20
Converted Double: 20.0

Answer

```
import java.io.*;
import java.util.*;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int x = sc.nextInt();
      double db = x;
      System.out.println("Original integer: "+x+"");
      System.out.println("Converted double: "+db+"");

   }

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 1_Q4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Vishal and Arun are discussing the properties of numbers. Vishal gives Arun two integers. He asks Arun to check if the sum of these two numbers is a multiple of their product.

Can you assist Arun and determine whether the sum is a multiple of the product?

Input Format

The input consists of two space-separated integers.

Output Format

The output prints:

1. "Sum is Multiple of Product" if the sum of the two numbers is divisible by their product.
2. "Sum is Not Multiple of Product" otherwise.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1 2

Output: Sum is Not Multiple of Product

Answer

```
import java.io.*;
import java.util.*;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int a = sc.nextInt();
      int b = sc.nextInt();
      int sum = a+b;
      int product = a*b;
      if(sum==product)
      {
         System.out.println("Sum is Multiple of Product");
      }
      else
      System.out.println("Sum is not Multiple of Product");
   }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 1_Q5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement:

Emily has a beautiful circular garden in her backyard. She's interested in calculating two important measurements for her garden: the circumference and the area. To do this, she needs a program that can take the radius of her circular garden as input and provide the calculated circumference and area as output. The formulas she should use are as follows:

To calculate the circumference (C) of a circle, you can use the formula:

C = 2 * π * r

A = π * r^2

Where:

C represents the circumference.

A represents the area.

π (pi) is approximately 3.14159.

r is the radius of the circle.

Emily is not a programmer, and she needs your help to create a program that will make these calculations for her garden.

Input Format

The first line of input contains a single double-point number radius, representing the radius of the circle.

Output Format

The output should consist of two lines:

The first line should print the circumference of the circle rounded to 2 decimal places, followed by the unit "meters".

The second line should print the area of the circle rounded to 2 decimal places, followed by the unit "square meters".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3.0

Output: Circumference: 18.85 meters
Area: 28.27 square meters

Answer

```
import java.io.*;
import java.util.*;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      double r = sc.nextDouble();
      double c = 2*3.14159*r;
```

```
        double a = 3.14159*r*r;
        System.out.printf("Circumference: %.2f meters",c);
        System.out.printf("Area: %.2f square meters" ,a);
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 1_Q6

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Joey is learning about bitwise operations and is working on a project that involves extracting specific bits from integers. He needs to write a program that takes an integer and the number of bits N as input and outputs the value of the lowest N bits of the integer.

Help Joey in his project to understand and visualize how bitwise operations work in practical scenarios.

Input Format

The first line of input consists of an integer X, representing the given integer.

The second line consists of an integer N, representing the number of bits to extract.

Output Format

The output displays "Result: " followed by an integer representing the value of the lowest N bits of the given integer.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 85
2
Output: Result: 1

Answer

```
import java.io.*;
import java.util.*;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
       int X = sc.nextInt();
       int N = sc.nextInt();
       int mask = (1<<N) - 1;
       int result = X & mask;
       System.out.println("Result: "+result);
   }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 1_Q7

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement:

Miles is working on a program that involves analyzing two integers. He wants to check if either one of the integers is both:

Less than or equal to zero, andOdd.Can you help him create a program that identifies whether either of the integers meets these conditions?

Input Format

The input consists of two integers on separate lines, denoted as 'input1' and 'input2'.

Output Format

A single line with a boolean result (either 'true' or 'false') indicating whether either 'input1' or 'input2' is both less than or equal to zero and odd.

Refer to the sample output for format specifications

Sample Test Case

Input: -45
10
Output: true

Answer

```
import java.io.*;
import java.util.*;
class main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
         int input1 = sc.nextInt();
         int input2 = sc.nextInt();
         if(((input1%2!=0)&&(input1<=0)) || ((input2%2!=0)&&(input2<=0)))
         {
           System.out.println("true");
         }
         else
         System.out.println("false");
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 1_Q8

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In the Kingdom of Finance, the royal treasury is managed by the treasurer, Sir Cedric. Sir Cedric tracks the daily expenses of the kingdom using an expense report that lists three major categories: food, clothing, and utilities. However, the King wants to know if the average daily expense is greater than at least two of these categories to ensure the kingdom is spending wisely.

Your task is to help Sir Cedric determine if the average daily expense is greater than two of the categories. Specifically, you need to calculate the average of the three expenses and check if it is greater than any two categories.

Note: Use the ternary operator

Input Format

Three integers a, b, and c represent the daily expenses for food, clothing, and utilities. Each integer is provided on a single line.

Output Format

The average of the three expenses, rounded to two decimal places.

A message indicating whether the average is greater than at least two of the expense categories.

1. If the average is greater than the two smallest monthly expenses, print "Average is greater than both X and Y," where X and Y are the two smallest expenses.
2. Otherwise, display "Average is not greater than two smallest expenses".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
6
10

Output: 6.67
Average is greater than both 4 and 6

Answer

```
import java.io.*;
import java.util.*;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int a = sc.nextInt();
      int b = sc.nextInt();
      int c = sc.nextInt();
      double average = (a+b+c)/3.0;
      System.out.printf("%.2f",average);
      if((average>a && average>b))
      System.out.println("\nAverage is greater than both "+a+" and "+b+"");
      else if(((average>b) && (average>c)))
```

```
        System.out.println("\nAverage is greater than both "+b+" and "+c+"");
        else if(((average>a)&&(average>c)))
        System.out.println("\nAverage is greater than both "+a+" and "+c+"");
        else
        System.out.println("\nAverage is not greater than two smallest expenses");


    }


}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 1_Q9

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Phill is a quality control manager at a manufacturing plant. He needs to verify if a sensor reading at a midpoint station (S2) falls exactly halfway between the readings of the previous station (S1) and the next station (S3). Help him by developing a program that checks if the second sensor reading is the average (midpoint) of the first and third sensor readings.

Use the relational operator to solve the program.

Input Format

The first line of input consists of an integer S1, representing the sensor reading of the first station.

The second line consists of an integer S2, representing the sensor reading of the midpoint station.

The third line consists of an integer S3, representing the sensor reading of the next station.

Output Format

The first line of output displays a boolean value representing whether the sensor reading at the midpoint station is halfway between the readings of the first and the next stations.

The second line displays one of the following:

1. If the result is true, print "The second integer is halfway between the first and third integers."
2. Otherwise, print "The second integer is not halfway between the first and third integers."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
7
10

Output: false
The second integer is not halfway between the first and third integers.

Answer

```
import java.io.*;
import java.util.*;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int S1 = sc.nextInt();
      int S2 = sc.nextInt();
      int S3 = sc.nextInt();
      int average = (S1+S3)/2;
      if(average == S2){
         System.out.println("true");
         System.out.println("The second integer is halfway between the first and
third integers.");
```

```
        }
        else{
            System.out.println("false");
            System.out.println("The second integer is not halfway between the first
and third integers.");

        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 1_Q10

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Aishu is supervising a construction project that needs to be completed with the help of three workers: A, B, and C.

She knows how many days each of them would take to complete the entire project individually:

A can complete it in x days,B in y days,C in z days.

Initially, all three workers (A, B, and C) work together for d1 days.

After that, C leaves, and only A and B continue for another d2 days.

Then B also leaves, and A works alone to finish the remaining work.

Your tasks is to help aishu to implement this functionality using the class WorkDistribution and Method calculateWork(int x, int y, int z, int d1, int d2)

Calculate the total work completed in the first d1 days by A, B, and C.Calculate the work completed in the next d2 days by A and B.Determine the remaining work after these d1 + d2 days.

Input Format

The first line of input contains five space-separated integers: x y z d1 d2

where:

x represents the Days A takes to complete the work alone

y represents the Days B takes to complete the work alone

z represents the Days C takes to complete the work alone

d1 represents the Days A, B, and C work together

d2 represents the Days A and B work together (after C leaves)

Output Format

The first line of output prints "Work done in first d1 days (A+B+C): " followed by a double value rounded to 2 decimal places.

The second line of output prints "Work done in next d2 days (A+B): " followed by a double value rounded to 2 decimal places.

The third line prints "Remaining work: " followed by a double value rounded to 2 decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10 20 30 2 2

Output: Work done in first d1 days (A+B+C): 0.37
Work done in next d2 days (A+B): 0.30
Remaining work: 0.33

Answer

```
import java.util.Scanner;
import java.io.*;
class WorkDistribution{
    public void CalculateWork(int x, int y, int z, int d1, int d2){
        double rateA = 1.0/x;
        double rateB = 1.0/y;
        double rateC = 1.0/z;

        double workDoneD1 = ((rateA + rateB + rateC)*d1);
        double workDoneD2 = ((rateA + rateB)*d2);
        double remainingWork = (1.0 - (workDoneD1 + workDoneD2));
        String D1 = String.format("%.2f", workDoneD1);
        String D2 = String.format("%.2f", workDoneD2);
        String R = String.format("%.2f", remainingWork);



        System.out.println("Work done in first d1 days(A+B+C): "+D1+"");
        System.out.println("Work done in next d2 days(A+B): "+D2+"");
        System.out.println("Remaining work: "+R+"");
    }
    public static void main(String[] args){
        Scanner scanner = new Scanner(System.in);
        int x = scanner.nextInt();
        int y = scanner.nextInt();
        int z = scanner.nextInt();
        int d1 = scanner.nextInt();;
        int d2 = scanner.nextInt();
        WorkDistribution wd = new WorkDistribution();
        wd.CalculateWork(x,y,z,d1,d2);
        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 1_PAH

Attempt : 1
Total Mark : 40
Marks Obtained : 35

Section 1 : Coding

1. PROBLEM STATEMENT:

Maria, a software developer, is working on a program to determine if two given integers which can be either positive or negative integers have the same parity (both even or both odd). She needs your help in writing this program.

Write a program that takes two integers as input and checks if both integers are either even or odd.

Input Format

The input consists of two lines:

The first line consists of an integer (input1) which can be either positive or negative.

The second line consists of an integer (input2) which can be either positive or negative.

Output Format

The output is dispalyed in the following format:

If both integers have the same parity (i.e., both even or both odd), print:

"Both integers are either even or odd"

Otherwise, print:

"The integers have different parities"

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 2
-4
Output: Both integers are either even or odd

Answer

```
import java.io.*;
import java.util.Scanner;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int m = sc.nextInt();
      int n = sc.nextInt();
      if((m%2==0) && (n%2==0)){
         System.out.println("Both integers are either even or odd");
      }
      else if((m%2!=0)&&(n%2!=0)){
```

```
            System.out.println("Both integers are either even or odd");
        }
        else{
            System.out.println("The integers have different parities");
        }
    }
}
```

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

2. Problem Statement

In the Kingdom of Delivery Logistics, there is a giant truck used for transporting packages across the kingdom. The truck has a maximum capacity represented by an integer, and each package also has a specific weight. The truck's efficiency and safety depend on whether the weight of the package is below a certain threshold.

The kingdom's delivery service has a rule: if the weight of a package is less than one-third of the truck's total capacity, the package is eligible for quick processing and dispatch. However, if the weight is too heavy, the package will require special handling.

As a logistics manager, you need to check whether the weight of the package is less than one-third of the truck's total capacity.

Write a program using a ternary operator that helps determine whether the package weight meets the requirement for quick processing or if it needs special handling.

Input Format

The first line of input consists of an integer p, representing the weight of the package.

The second line consists of an integer w, representing the total weight capacity of the truck.

Output Format

The first line of output prints "One-third of Truck: X," where X is one-third of the

truck's total weight capacity as a double value with two decimal places.

The second line of output displays one of the following:

1. If p is less than one-third of the truck's total weight capacity, print "Package weight is less than one-third of the truck's capacity".
2. Otherwise, print "Package weight is not less than one-third of the truck's capacity".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 13
60
Output: One-third of Truck: 20.00
Package weight is less than one-third of truck's capacity

Answer

```
import java.io.*;
import java.util.Scanner;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int p = sc.nextInt();
      int w = sc.nextInt();
      double t = w/3;
      System.out.printf("One-third of Truck: %.2f\n",t);
      if(w/3 > p){
         System.out.println("Package weight is less than one-third of truck's
capacity");
      }
      else if((w/3 < p)|| (w/3 == p)){
         System.out.println("Package weight is not less than one-third of truck's
capacity");
      }
   }
}
```

Status : Partially correct ***Marks : 8.5/10***

3. PROBLEM STATEMENT:

Maria, a software developer, is working on a project to create a simple program to determine which of two integers is closest to zero. The integers can be either positive or negative. The program needs to take two integer inputs and calculate which one is closer to zero. If both integers are equidistant from zero, the program should return 0.

Input Format

The input contains two lines:

The first line of the input contains an integer, which can be either a positive or a negative integer.

The second line of the input contains an integer, which can be either a positive or a negative integer.

Output Format

The output displays the integer that is closest to zero in the following format:

"The integer closest to zero is: [closest_integer]"

Here, [closest_integer] should be replaced with the integer that is closer to zero based on its absolute value.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
8
Output: The integer closest to zero is: 5

Answer

```
import java.io.*;
import java.util.Scanner;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int m = sc.nextInt();
      int n = sc.nextInt();
      if((m>0)&&(n>0)){
         if(m<n){
            System.out.println("The integer closest to zero is: "+m);
         }
         else{
            System.out.println("The integer closest to zero is: "+n);
         }
      }
      else if((m<0)&&(n<0)){
         if(m<n){
            System.out.println("The integer closest to zero is: "+n);
         }
         else{
            System.out.println("The integer closest to zero is: "+m);
         }
      }
      else if((m<0) && (n>0)){
         int M = -m;
         if(M<n){
            System.out.println("The integer closest to zero is: "+n);
         }
         else{
            System.out.println("The integer closest to zero is: "+m);
         }
      }
      else if((m>0)&&(n<0)){
         int N = -n;
         if(m<N){
         System.out.println("The integer cloest  to zero is: "+m);
         }
         else{
            System.out.println("The integer closest to zero is: "+n);
         }
      }
```

```
        else if(m==n){
            System.out.println("0");
        }
        else if((m==0) || (n==0)){
            System.out.println("The integer closest to zero is: 0");
        }

    }
}
```

Status : Partially correct **Marks : 6.5/10**

4. Problem Statement

Mickey and Miney are walking through a magical forest. The forest is full of enchanted stones, each with a unique number. There is a legend that says the magic power of the stones can be revealed by using a special operation. To determine the magic power of a given stone, you need to perform a bitwise AND operation with the number 15.

Each stone's number is represented by an integer, and Mickey needs to find the magic power of each stone by applying this operation.

Your task is to help Mickey compute the result of the bitwise AND operation of the given stone number with 15, and print the result.

Input Format

The input consists of a single integer.

Output Format

The output should display a single integer, which is the result of the bitwise AND operation between input and 15.

Refer to the sample output for format specifications.

Sample Test Case

Input: 25

Output: 9

Answer

```
import java.io.*;
import java.util.Scanner;
class main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int m = n & 15;
        System.out.println(m);
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 1_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

In the faraway land of Arithmetica, there exists an ancient calculator that can only perform bitwise operations. The calculator is locked with a secret code that only works when the number is modified using a special operation called right shifting.

The ruler of Arithmetica, King Thales, needs your help to unlock the calculator. The lock on the calculator is encoded with a number, and the calculator will only open if you apply a right shift by 2 on the number. Your task is to help King Thales determine the magic number that will unlock the ancient calculator.

Input Format

The first line of input represents an integer.

Output Format

The output should display the right-shifted value by 2 bits.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 16

Output: 4

Answer

```
import java.io.*;
import java.util.Scanner;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      int m = n>>2;
      System.out.println(m);
   }
}
```

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

2. Problem Statement:

"Write a program that helps identify the type of a triangle based on the lengths of its three sides. The program prompts the user to input the lengths of sides 'a,' 'b,' and 'c,' and then it classifies the triangle as 'Equilateral' if all sides are equal, 'Isosceles' if two sides are equal, or 'Scalene' if all sides are different. Can you provide the Java code for this task?"

Input Format

The first line of the input is an integer 'a' representing the length of side 'a.'

The second line of the input is an integer 'b' representing the length of side 'b.'

The third line of the input is an integer 'c' representing the length of side 'c.'

Output Format

The program outputs a single line that specifies the type of the triangle: "Equilateral," "Isosceles," or "Scalene."

Sample Test Case

Input: 3
4
5
Output: The triangle is Scalene

Answer

```
import java.io.*;
import java.util.Scanner;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int a = sc.nextInt();
      int b = sc.nextInt();
      int c = sc.nextInt();
      if((a==b)&&(a==c)){
         System.out.println("The triangle is Equilateral");
      }
      else if((a==b) || (a==c) || (b==c)){
         System.out.println("The triangle is Isosceles");
      }
      else{
         System.out.println("The triangle is Scalene");
      }
   }
}
```

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

3. PROBLEM STATEMENT:

Jule a mathematician expert is given two integers to find if the second integer is above the average of the first and second integer. Write a

program that achieves this using the ternary operator.

Input Format

The first line of input represents the first integer.

The second line of input represents the second integer.

Output Format

The output should be displayed as "Below Average" or "Above Average"

REFER THE SAMPLE TESTCASES FOR THE FORMAT SPECIFICATIONS.

Sample Test Case

Input: 1
1
Output: Below Average

Answer

```
import java.io.*;
import java.util.Scanner;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int a = sc.nextInt();
      int b = sc.nextInt();
      int av = (a+b)/2;
      if(b>av){
         System.out.println("Above Average");
      }
      else{
         System.out.println("Below Average");
      }

   }
}
```

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

4. Problem Statement

In a logistics company, each delivery pack contains a specific number of items, and the priority customer receives double the amount. Write a program to determine the total number of delivery packs required for the operation, considering the number of items per pack and the number of customers given as input by the user.

Example

Input:

Number of items per pack = 96

Number of customers = 8

Output:

10

Explanation:

Given the number of items per pack = 96 and the number of customers = 8, the calculations are as follows:

Total number of items needed = number of items per pack * number of customers = 96 * 8 = 768.Priority customer's share = double the amount of items per pack = 2 * 96 = 192.Total items with the priority customer = total items needed + priority share = 768 + 192 = 960.Number of packs needed = (960 + 96 − 1)/ 96 = 10.98Since we cannot have a fraction of a pack, the output is 10.

Input Format

The input consists of two space-separated integers N and C, representing the number of items per pack and the number of customers.

Output Format

The output displays an integer, representing the total number of delivery packs required for the operation.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1 1

Output: 3

Answer

```
import java.io.*;
import java.util.Scanner;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int N = sc.nextInt();
      int C = sc.nextInt();
      int I = N*C;
      int S = 2*N;
      int T = I+S;
      int P = (T+N-1)/N;
      System.out.println(P);

   }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 2_Q1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Arun is working on a project to automate the process of determining whether a student has passed or failed based on their subject marks.

He aims to create a simple program that takes positive integers as marks for five subjects from the user. If the average of the marks is greater than or equal to 50, the student has passed the exam. Otherwise, the student has failed.

Help Arun to implement the project.

Input Format

The input consists of five space-separated integers, representing the marks in five subjects.

Output Format

The first line of output prints "Average score: " followed by an integer representing the average score.

The second line prints one of the following:

1. If the condition is satisfied, print "The student has passed".
2. Otherwise, the output prints "The student has failed".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 50 60 70 80 90

Output: Average score: 70
The student has passed

Answer

```
import java.io.*;
import java.util.Scanner;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int a = sc.nextInt();
      int b = sc.nextInt();
      int c = sc.nextInt();
      int d = sc.nextInt();
      int e = sc.nextInt();
      int A = (a+b+c+d+e)/5;
      System.out.println("Average score: "+A);
      if(A>=50){
         System.out.println("The student has passed");
      }
      else{
         System.out.println("The student has failed");
      }
   }
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 2_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Samantha is a diligent math student who is exploring the world of programming. She is learning Java and has recently studied conditional statements. One day, her teacher gives her an interesting problem to solve, which takes a number as input and checks whether it is a multiple of 5 or 7.

Help her complete the task.

Input Format

The input consists of a single integer N, representing the number to be checked.

Output Format

If the number is a multiple of 5 but not 7, the output prints "N is a multiple of 5".

If the number is a multiple of 7, the output prints "N is a multiple of 7".

Otherwise the output prints "N is neither multiple of 5 nor 7" where N is an entered integer.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10
Output: 10 is a multiple of 5

Answer

```
import java.io.*;
import java.util.Scanner;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int N = sc.nextInt();
      if((N%5==0)&&(N%7!=0)){
         System.out.println(""+N+" is a multiple of 5");
      }
      else if((N%7==0)&&(N%5!=0)){
         System.out.println(""+N+" is a multiple of 7");
      }
      else{
         System.out.println(""+N+" is neither multiple of 5 nor 7");
      }
   }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 2_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John is a fitness trainer, and he wants to use the BMI calculator to assess the body mass index of his clients. He has a list of clients based on their height and weight.

John plans to write a program to quickly determine the BMI and provide a classification for each client.

If BMI is less than 18.5, the program will classify it as "Underweight"If BMI is between 18.6 and 24.9, the program will classify it as "Normal Weight"If BMI is between 25.0 and 29.9, the program will classify it as "Overweight"If BMI is 30.0 or higher, the program will classify it as "Obese"

Note: Formula to calculate BMI = weight/(height*height)

Input Format

The first line of input consists of a double value, representing the height of the person in meters.

The second line consists of a double value, representing the weight of the person in kilograms.

Output Format

The first line of output prints "BMI: " followed by a double (rounded to two decimal places) representing the calculated BMI.

The second line prints "Classification: " followed by a string indicating the BMI category (Underweight, Normal Weight, Overweight, or Obese).

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1.2
45.2

Output: BMI: 31.39
Classification: Obese

Answer

```
import java.util.Scanner;
class main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        double height = sc.nextDouble();
        double weight = sc.nextDouble();
        double Bmi = weight/(height*height);
        System.out.printf("BMI: %.2f", Bmi);
        if(Bmi<18.5){
            System.out.println("Classification: Underweight");
        }
        else if(Bmi<=24.9 && Bmi>=18.6){
            System.out.println("Classification: Normal Weight");
        }
        else if(Bmi<=29.9 && Bmi >=25.0){
            System.out.println("Classification: Overweight");
        }
```

```
        else{
            System.out.println("Classification: Obese");
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 2_Q4

Attempt : 1
Total Mark : 10
Marks Obtained : 9

Section 1 : Coding

1. Problem Statement

Amit wants to evaluate the depreciation of his car over time to understand its current value and categorize it based on that value.

Write a program that helps him determine the current value of his car after a certain number of years of depreciation and classify it into one of three categories:

High: If the current value is greater than 10,000.Medium: If the current value is between 5,000 and 10,000, both inclusive.Low: If the current value is less than 5,000.

The depreciation rate of the car is 15% per year. The program should calculate the current value of the car after applying this depreciation over the given number of years and print the current value along with the category.

Input Format

The first line of input consists of an integer, representing the initial cost of the car.

The second line consists of an integer, representing the number of years the car has been depreciating.

Output Format

The first line of output prints a double value, representing the current value of the car, rounded off to two decimal places "Current Value: <value>".

The second line prints its category "Category: <categories>".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 20000
5
Output: Current Value: 8874.11
Category: Medium

Answer

```
import java.util.Scanner;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      double initialCost = sc.nextDouble();
      int years = sc.nextInt();
      sc.close();
      double depreciationRate = 0.15;
      double currentValue = initialCost*Math.pow(0.85, years);
      currentValue = Math.round(currentValue*100.0)/100.0;

      String category;
      if (currentValue > 10000){
         category = "High";
      }
      else if(currentValue >= 5000 && currentValue <= 10000){
```

```
            category = "Medium";
        }
        else{
            category = "Low";
        }
        System.out.println("Current Value: " + String.format("%.2f", currentValue));
        System.out.println("Category: "+ category+"");
    }
}
```

Status : Partially correct **Marks : 9/10**

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 2_Q5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ted, the computer science enthusiast, has accepted the challenge of writing a program that checks if the number of digits in an integer matches the sum of its digits.

Guide Ted in designing and writing the code to solve this problem using a 'do-while' loop.

Input Format

The input consists of an integer N, representing the number to be checked.

Output Format

If the sum is equal to the number of digits, print "The number of digits in N matches the sum of its digits."

Else, print "The number of digits in N does not match the sum of its digits."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 20
Output: The number of digits in 20 matches the sum of its digits.

Answer

```
import java.util.Scanner;
class Solution{
   public static void main(String[] args){
      Scanner scanner = new Scanner(System.in);
      int n = scanner.nextInt();
      int originalNumber = n;
      int digitCount = 0;
      int sumOfDigits = 0;
      if(n==0){
         digitCount = 1;
      }else{
         do{
            digitCount++;
            sumOfDigits += n%10;
            n/=10;
         }while(n!=0);
      }
      if(digitCount == sumOfDigits) {
         System.out.println("The number of digits in "+originalNumber+" matches
the sum of its digits.");
      }else{
         System.out.println("The number of digits in "+originalNumber+" does not
match the sum of its digits.");
      }
   }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 2_Q6

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Maya, a student in an arts and crafts class, wants to create a pattern using stars (*) in a specific format. She plans to use a program to help her construct the pattern.

Write a program that takes an integer as input and constructs the following pattern using nested for loops.

Input: 5

Output:

*
* *

```
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

Input Format

The input consists of a number (integer) representing the number of rows.

Output Format

The output displays the required pattern.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5

Output:
```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

Answer

```
import java.util.Scanner;
class Solution{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int rows = sc.nextInt();
```

```
        for(int i=1; i<=rows; i++){
            for(int j=1; j<=i; j++){
                System.out.print("* ");
            }
            System.out.println();
        }
        for(int i = rows - 1; i>=1; i--){
            for(int j=1; j<=i; j++){
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 2_Q7

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are taking part in a coding challenge where your task is to design a program that conjures a mesmerizing numerical pyramid pattern. The enchanting pattern is fashioned using a for loop and is customized based on user input.

Participants are prompted to unveil the pyramid's magic by specifying its height - essentially dictating the number of rows in this spellbinding creation.

Write a program that employs to weave this captivating numerical pyramid as shown below.

Example

Input:
4

Output:

Input Format

The input consists of a positive integer n representing the number of rows in the pattern.

Output Format

The output prints the required pyramid pattern, as shown in the sample output.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4

Output: 1
 123
 12345
1234567

Answer

```
import java.util.Scanner;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int rows = sc.nextInt();
      for(int i = 1; i<=rows; i++){
         for(int j=1; j<=(2*i)-1; j++){
            System.out.print(j);
         }
         System.out.println();
      }
   }
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 2_Q8

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

A bank generates secure codes using 3-digit numbers where each digit is unique, and the code must be divisible by 3. You are tasked with generating the first N such codes based on user input, ensuring the digits are unique and the number is divisible by 3.

Note: Use nested for loops to solve.

Input Format

The first line contains an integer N representing the number of valid codes to generate.

Output Format

The output prints N line,s each line contains a valid 3-digit code.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

Output: 102
105
108
120
123

Answer

```
import java.util.Scanner;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int N= sc.nextInt();
      sc.close();
      int count = 0;
      for(int i = 100; i<=999; i++){
         if(count == N){
            break;
         }
         int digit1 = i/100;
         int digit2 = (i/10)%10;
         int digit3 = i%10;
         if(digit1 != digit2 && digit2 != digit3 && digit1 != digit3){
            if(i%3==0){
               System.out.println(i);
               count++;
            }
         }
      }
   }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 2_PAH

Attempt : 1
Total Mark : 40
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

You are given a number of distribution centers (rows) and are tasked with generating a zigzag shipment route pattern. Each shipment route should alternate between left-to-right and right-to-left, as described below.

The program should print the zigzag pattern with a tab (\t) separating the columns. For each row, the shipment numbers should follow a diagonal pattern where numbers are placed in a zigzag, left to right on odd rows and right to left on even rows.

Input Format

The input consists of an integer N, which represents the number of distribution centers (rows) for the zigzag pattern.

Output Format

The output prints the zigzag pattern with N rows, formatted with a tab space (\t) separating the columns.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

Output:	1
	2	6
	3	7	10
	4	8	11	13
5	9	12	14	15

Answer

-

Status : Skipped ***Marks : 0/10***

2. Problem Statement

Ravi wants to estimate the total utility bill for a household based on the consumption of electricity, water, and gas.

Write a program to calculate the total bill using the following criteria:

The cost per unit for electricity is 0.12, for water is 0.05, and for gas is 0.08.A discount is applied to the total cost based on the following conditions:If the total cost is 100 or more, a 10% discount is applied.If the total cost is between 50 and 99.99, a 5% discount is applied.No discount is applied if the total cost is less than 50.

The program should output the total bill after applying the discount with two decimal places.

Input Format

The input consists of three double values, representing the number of units

consumed for electricity, water, and gas respectively.

Output Format

The output prints a double value, representing the total bill after applying the discount, formatted to two decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1000.0
200.0
100.0
Output: 124.20

Answer

```
import java.util.Scanner;
class Main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      double e = sc.nextDouble();
      double w = sc.nextDouble();
      double g = sc.nextDouble();
      double total = e * 0.12 + w * 0.05 + g * 0.08;
      if(total >= 100) total -= total * 0.10;
      else if(total >= 50) total -= total * 0.05;
      System.out.printf("%.2f", total);
   }
}
```

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

3. Problem Statement

Sampad is a high school teacher who wants to convert numeric grades into letter grades.

Write a program that accepts a numeric grade as input. The program

should then convert this numeric grade into a letter grade based on specific conditions. The letter grades are A, B, C, D and F.

The conversion is determined by the following conditions:

If the numeric grade is 90 or higher, it's an "A"If the numeric grade is between 80 and 89 (inclusive), it's a "B"If the numeric grade is between 70 and 79 (inclusive), it's a "C"If the numeric grade is between 60 and 69 (inclusive), it's a "D"If the numeric grade is below 60, it's an "F"

Input Format

The input consists of an integer representing the numeric grade of the student.

Output Format

The output prints the letter grade corresponding to the input numeric grade as "Letter Grade: <grade>".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 85

Output: Letter Grade: B

Answer

```
import java.util.Scanner;
class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int grade = sc.nextInt();
        String result;
        if(grade >= 90) result = "A";
        else if(grade >=80) result = "B";
        else if(grade >= 70)result = "C";
        else if(grade >= 60) result = "D";
        else result = "F";
        System.out.println("Letter Grade: "+ result);
    }
}
```

Status : Correct *Marks : 10/10*

4. Problem Statement

Rohit is tasked with designing a program to analyze the digits of a given integer.

Write a program to help Rohit that takes an integer as input and identifies the minimum odd digit and the maximum even digit present in the number. If no odd or even digits are present, display appropriate messages.

Implement the solution using a 'while' loop to iterate through the digits of the given number.

Input Format

The input consists of an integer n.

Output Format

The first line of output prints the message "Minimum odd digit: " followed by an integer representing the smallest odd digit found in the input number.

If no odd digit exists, it prints "There are no odd digits in the number."

The second line of output prints the message "Maximum even digit: " followed by an integer representing the largest even digit found in the input number.

If no even digit exists, it prints "There are no even digits in the number."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3465
Output: Minimum odd digit: 3
Maximum even digit: 6

Answer

```
import java.util.Scanner;
class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt(), minOdd = 9, maxEven = -1, d;
        while(n > 0){
            d = n%10;
            if(d%2 == 0 && d > maxEven) maxEven = d;
            if(d%2 == 1 && d < minOdd) minOdd = d;
            n /= 10;
        }
        if(minOdd == 9)
        System.out.println("There are no odd digits in the number.");
        else
        System.out.println("Minimum odd digit: "+ minOdd);
        if(maxEven == -1)
        System.out.println("There are no even digits in the number.");
        else
        System.out.println("Maximum even digit: "+ maxEven);

    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 2_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Joe has a favourite number, let's call it X. He wants to check if X is divisible by the sum of its digits. If it is, he considers it a lucky number. If not, he wants to find the closest smaller number, that is divisible by the sum of digits of X. Joe has challenged his friends to solve this puzzle at his birthday party.

Example

Input:

157

Output:

157 is not divisible by the sum of its digits.

The closest smaller number that is divisible: 156

Explanation:

The sum of the digits of X is 1+5+7=13. Since 157 is not divisible by 13, we need to find the closest smaller number that is divisible by 13. 156 is divisible by 13, it is the closest smaller number that meets the requirement.

Input Format

The input consists of an integer X, representing Joe's favourite number.

Output Format

If X is a lucky number, then the output must be in the format: "X is divisible by the sum of its digits."

If not, then the output must be in the format:

"X is not divisible by the sum of its digits.

The closest smaller number that is divisible: Y",

where X is the entered number and Y is the closest number.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 120

Output: 120 is divisible by the sum of its digits.

Answer

```
import java.util.*;

class LuckyNumberCheck {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int X = sc.nextInt();
      int sum = 0, temp = X;
      while (temp > 0) {
         sum += temp % 10;
```

```
        temp /= 10;
      }
      if (X % sum == 0) {
        System.out.println(X + " is divisible by the sum of its digits.");
      } else {
        int Y = X - 1;
        while (Y > 0 && Y % sum != 0) {
          Y--;
        }
        System.out.println(X + " is not divisible by the sum of its digits.");
        System.out.println("The closest smaller number that is divisible: " + Y);
      }
    }
}
```

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

2. Problem Statement

Raj is solving a physics problem involving projectile motion, where he needs to calculate the time a ball hits the ground using a quadratic equation of the form $ax^2 + bx + c = 0$. Depending on the coefficients, the ball may hit the ground once, twice, or not at all in real time.

Help Raj find all real roots of the equation, if any.

Note: discriminant = b2 - 4ac

Input Format

The input consists of three space-separated doubles a, b, and c, representing the coefficients of the quadratic equation.

Output Format

If there are two real roots, print:

- "Two real solutions:"
- "Root1 = <value>"
- "Root2 = <value>"

If there is one real root, print:

- "One real solution:"
- "Root = <value>"

If there are no real roots, print:

- "There are no real solutions."

Note: values are rounded to two decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1 6 9

Output: One real solution:
Root = -3.00

Answer

```
import java.util.*;
class Main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      double a = sc.nextDouble();
      double b = sc.nextDouble();
      double c = sc.nextDouble();
      double d = b*b-4*a*c;
      if(d>0){
         double r1 = (-b+Math.sqrt(d))/ (2*a), r2 = (-b-Math.sqrt(d))/ (2*a);
         System.out.printf("Two real solutions:%nRoot1 = %.2f%nRoot2 = %.2f", r1,
r2);
      }
      else if(d==0){
         double r = -b/(2*a);
         System.out.printf("One real solution:%nRoot = %.2f",r);
      }else System.out.print("There are no real solutions.");

   }
}
```

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

3. Problem Statement

Maya, a student in an arts and crafts class, wants to create a pattern using stars (*) in a specific format. She plans to use a program to help her construct the pattern.

Write a program that takes an integer as input and constructs the following pattern using nested for loops.

Input: 5

Output:

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

Input Format

The input consists of a number (integer) representing the number of rows.

Output Format

The output displays the required pattern.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5

Output: *

```
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

Answer

```
import java.util.*;
class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        for(int i = 1; i<=n; i++){
            for(int j=1; j<=i; j++)
            System.out.print("* ");
            System.out.println();
        }
        for(int i=n-1; i>=1; i--){
            for(int j=1; j<=i;j++)
            System.out.print("* ");
            System.out.println();
        }

    }
}
```

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

4. Problem Statement

John is a fitness trainer, and he wants to use the BMI calculator to assess the body mass index of his clients. He has a list of clients based on their height and weight.

John plans to write a program to quickly determine the BMI and provide a classification for each client.

If BMI is less than 18.5, the program will classify it as "Underweight"If BMI is between 18.6 and 24.9, the program will classify it as "Normal Weight"If BMI is between 25.0 and 29.9, the program will classify it as "Overweight"If BMI is 30.0 or higher, the program will classify it as "Obese"

Note: Formula to calculate BMI = weight/(height*height)

Input Format

The first line of input consists of a double value, representing the height of the person in meters.

The second line consists of a double value, representing the weight of the person in kilograms.

Output Format

The first line of output prints "BMI: " followed by a double (rounded to two decimal places) representing the calculated BMI.

The second line prints "Classification: " followed by a string indicating the BMI category (Underweight, Normal Weight, Overweight, or Obese).

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1.2
45.2

Output: BMI: 31.39
Classification: Obese

Answer

```
import java.util.*;
class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        double height = sc.nextDouble();
```

```
        double weight = sc.nextDouble();
        double bmi = weight/ (height*height);
        System.out.printf("BMI: %.2f\n", bmi);
        if(bmi < 18.5)
        System.out.println("Classification: Underweight");
        else if(bmi >=18.6 && bmi <= 24.9)
        System.out.println("Classification: Normal Weight");
        else if(bmi >= 25.0 && bmi <= 29.9)
        System.out.println("Classification: Overweight");
        else
        System.out.println("Classification: Obese");
    }

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 3_Q1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Rosh is intrigued by numerical patterns. Today, she stumbled upon a puzzle while working with arrays. She wants to compute the sum of the third-largest and second-smallest elements from a list of integers. She seeks your help to implement a program that solves this for her efficiently.

Input Format

The first line of input is an integer N, representing the size of the array.

The second line of input consists of N space-separated integers, representing the elements of the array.

Output Format

The output displays a single integer representing the sum of the third-largest and second-smallest elements in the array.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 10
10 20 30 40 50 60 70 80 90 100
Output: 100

Answer

```
import java.util.*;

class ArrayPuzzle {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int[] arr = new int[N];
        for (int i = 0; i < N; i++) {
            arr[i] = sc.nextInt();
        }
        TreeSet<Integer> sortedSet = new TreeSet<>();
        for (int num : arr) {
            sortedSet.add(num);
        }
        List<Integer> sortedList = new ArrayList<>(sortedSet);
        if (sortedList.size() < 3) {
            System.out.println("Not enough unique elements");
        } else {
            int thirdLargest = sortedList.get(sortedList.size() - 3);
            int secondSmallest = sortedList.get(1);
            System.out.println(thirdLargest + secondSmallest);
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 3_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Monica is interested in finding a treasure but the key to opening is to get the sum of the main diagonal elements and secondary diagonal elements.

Write a program to help Monica find the diagonal sum of a square 2D array.

Note: The main diagonal of the array consists of the elements traversing from the top-left corner to the bottom-right corner. The secondary diagonal includes elements from the top-right corner to the bottom-left corner.

Input Format

The first line of input consists of an integer N, representing the number of rows and columns.

The following N lines consist of N space-separated integers, representing the 2D array elements.

Output Format

The first line of output prints "Sum of the main diagonal: " followed by an integer, representing the sum of the main diagonal.

The second line prints "Sum of the secondary diagonal: " followed by an integer, representing the sum of the secondary diagonal.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1 2 3
4 5 6
7 8 9

Output: Sum of the main diagonal: 15
Sum of the secondary diagonal: 15

Answer

```
import java.util.*;

class DiagonalSum {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int[][] matrix = new int[N][N];
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                matrix[i][j] = sc.nextInt();
            }
        }
        int mainDiagonalSum = 0;
        int secondaryDiagonalSum = 0;
        for (int i = 0; i < N; i++) {
            mainDiagonalSum += matrix[i][i];
            secondaryDiagonalSum += matrix[i][N - 1 - i];
        }
```

```
        System.out.println("Sum of the main diagonal: " + mainDiagonalSum);
        System.out.println("Sum of the secondary diagonal: " +
secondaryDiagonalSum);
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 3_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are developing a warehouse management system for a shipping company. The system uses an integer array to represent the weights of packages in a specific order. To verify that the weight capacity is not exceeded, the program needs to calculate the sum of the weights of the first and last packages in the list.

Task:

Write a code to calculate the sum of the weights of the first and last packages in the list. The program should take an integer array as input and return the total weight of the first and last packages.

Input Format

The first line of the input is an integer N representing the size of the array.

The second line of the input is N space-separated integer values.

Output Format

The output is displayed in the following format:

"Sum of the first and last elements: <<Sum>>"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50
Output: Sum of the first and last elements: 60

Answer

```
import java.util.*;

class PackageWeightSum {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int[] weights = new int[N];
        for (int i = 0; i < N; i++) {
            weights[i] = sc.nextInt();
        }
        int sum = weights[0] + weights[N - 1];
        System.out.println("Sum of the first and last elements: " + sum);
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 3_Q4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Sesha is developing a weather monitoring system for a region with multiple weather stations. Each weather station collects temperature data hourly and stores it in a 2D array.

Write a program that can add the temperature data from two different weather stations to create a combined temperature record for the region.

Input Format

The first line of input consists of two space-separated integers N and M, representing the number of rows and columns of the matrices, respectively.

The next N lines consist of M space-separated integers, representing the values of the first matrix.

The following N lines consist of M space-separated integers, representing the values of the second matrix.

Output Format

The output prints the addition of the two matrices in N rows and M columns, representing the combined temperature record.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3 3
1 2 3
4 5 6
7 8 9
1 1 1
2 2 2
3 3 3

Output: 2 3 4
6 7 8
10 11 12

Answer

```
import java.util.*;

class TemperatureMatrixAddition {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int M = sc.nextInt();
        int[][] matrix1 = new int[N][M];
        int[][] matrix2 = new int[N][M];
        int[][] result = new int[N][M];
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++) {
                matrix1[i][j] = sc.nextInt();
            }
        }
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++) {
```

```
            matrix2[i][j] = sc.nextInt();
        }
    }
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            System.out.print(result[i][j] + " ");
        }
        System.out.println();
    }
}
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 3_Q5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Sharon is creating a program that finds the first repeated element in an integer array. The program should efficiently identify the first element that appears more than once in the given array. If no such element is found, it should appropriately display a message.

Help Sharon to complete the program.

Input Format

The first line of input consists of an integer n, representing the number of elements in the array.

The second line consists of n space-separated integers, representing the array elements.

Output Format

If a repeated element is found, print the first element that appears more than once.

If no repeated element is found, print "No repeated element found in the array".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 8
12 21 13 14 21 36 47 21
Output: 21

Answer

```
import java.util.*;

class FirstRepeatedElement {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      int[] arr = new int[n];
      Set<Integer> seen = new HashSet<>();
      int repeated = -1;
      for (int i = 0; i < n; i++) {
         arr[i] = sc.nextInt();
      }
      for (int num : arr) {
         if (seen.contains(num)) {
            repeated = num;
            break;
         }
         seen.add(num);
      }
      if (repeated != -1) {
         System.out.println(repeated);
      } else {
         System.out.println("No repeated element found in the array");
      }
```

```
    }
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 3_PAH

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

In a customer loyalty program, reward points are logged in a sorted array as customers make transactions. Occasionally, due to system errors, duplicate entries for the same transaction may appear. To ensure accurate reward calculations, it's crucial to remove these duplicates from the list.

Write a program to process the array of reward points, removing any duplicates while preserving the order of unique entries. The program should then display the cleaned list of unique reward points and the total count of these unique points.

Input Format

The first line of input consists of an integer N, representing the number of reward points.

The second line consists of N space-separated integers, representing the reward points in sorted order.

Output Format

The first line of output prints the cleaned list of unique reward points separated by a space.

The second line of output prints an integer representing the total count of unique reward points.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
100 100 200
Output: 100 200
2

Answer

```
import java.util.*;
class Main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      int[] arr = new int[n];
      for(int i = 0; i<n; i++)
      arr[i] = sc.nextInt();
      LinkedHashSet<Integer> set = new LinkedHashSet<>();
      for(int x : arr) set.add(x);
      for(int x : set)
      System.out.print(x + " ");
      System.out.println();
      System.out.println(set.size());
   }
}
```

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

2. Problem Statement

Eminem is a billiard player who enjoys playing billiards and also likes solving mathematical puzzles. He notices that the billiard balls on the table are arranged in a grid, and he is curious to find the sum of the numbers written on each ball.

Write a program to find the sum of all the numbers written on each ball in the grid.

Input Format

The first line of input consists of an integer N, representing the number of rows.

The second line consists of an integer M, representing the number of columns.

The following lines N lines consist of M space-separated integers, representing the numbers written on each ball.

Output Format

The output prints an integer representing the sum of all the numbers written on each ball.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
3
1 2 3
4 5 6
7 8 9

Output: 45

Answer

```
import java.util.*;
public class Main{
    public static void main(String[] args){
```

```
        Scanner scanner = new Scanner(System.in);
        int N = scanner.nextInt();
        int M = scanner.nextInt();
        long sum = 0;
        for(int i=0; i<N; i++){
            for(int j = 0; j<M;j++){
                sum += scanner.nextInt();
            }
        }
        System.out.println(sum);
        scanner.close();
    }
}
```

Status : Correct *Marks : 10/10*

3. Problem Statement

Priya is building a system to automate image transformations using matrix operations. To do this, she needs to multiply two matrices representing pixel data and transformation rules.

Help Priya perform matrix multiplication and print the resulting matrix if the operation is valid.

Input Format

The first line of input consists of two int values, representing the number of rows R1 and columns C1 of the first matrix.

The next R1 × C1 integers represent the elements of the first matrix.

The next line consists of two int values, representing the number of rows R2 and columns C2 of the second matrix.

The next R2 × C2 integers represent the elements of the second matrix.

Output Format

If matrix multiplication is possible, print R1 lines, each containing C2 space-separated int values representing the resulting matrix.

Otherwise, print "Matrix multiplication not possible".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2 3
1 2 3
4 5 6
3 2
7 8
9 10
11 12
Output: 58 64
139 154

Answer

```
import java.util.Scanner;
class Main{
    public static void main(String[] args){
        Scanner scanner = new Scanner(System.in);
        int R1= scanner.nextInt();
        int C1= scanner.nextInt();
        int[][] matrix1 = new int[R1][C1];
        for(int i=0; i<R1;i++){
            for(int j = 0; j<C1;j++){
                matrix1[i][j] = scanner.nextInt();
            }
        }
        int R2= scanner.nextInt();
        int C2= scanner.nextInt();
        int[][] matrix2 = new int[R2][C2];
        for(int i =0; i<R2; i++){
            for(int j=0; j<C2;j++){
                matrix2[i][j] = scanner.nextInt();
            }
        }
        if(C1 != R2){
            System.out.println("Matrix multiplication not possible");
        }
```

```
        else{
            int[][] resultMatrix = new int[R1][C2];

            for(int i=0; i<R1;i++){
                for(int j=0; j<C2; j++){
                    for(int k=0; k<C1; k++){
                        resultMatrix[i][j] += matrix1[i][k] * matrix2[k][j];
                    }
                }
            }
            for(int i=0;i<R1;i++){
                for(int j=0; j<C2;j++){
                    System.out.print(resultMatrix[i][j] + (j==C2-1?"":""));
                }
                System.out.println();
            }
        }
        scanner.close();
    }
}
```

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

4. Problem Statement

Egath is participating in a coding hackathon, and one of the challenges requires him to work with an array of integers. The task is to remove exactly one element from the array such that the sum of the remaining elements is a prime number.

Help Egath find the first possible prime sum by removing one element or determining if no such prime sum can be achieved.

Input Format

The first line of input consists of an integer N, representing the number of elements in the array.

The second line consists of N space-separated integers, representing the array elements.

Output Format

If removing one element results in a prime sum, print the sum.

If no such prime sum can be achieved by removing exactly one element, print "No valid prime sum found".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1 2 3
Output: 5

Answer

```
import java.util.Scanner;
class Main{
   static boolean isPrime(long num){
      if(num<=1){
         return false;
      }
      for(long i=2; i*i<=num;i++){
         if(num%i == 0){
            return false;
         }
      }
      return true;
   }
   public static void main(String[] args){
      Scanner scanner = new Scanner(System.in);
      int N = scanner.nextInt();
      long[] arr = new long[N];
      long totalSum = 0;
      for(int i=0; i<N;i++){
         arr[i] = scanner.nextLong();
         totalSum += arr[i];
      }
      long firstPrimeSum = -1;
      for(int i=0; i<N;i++){
```

```
            long currentSum = totalSum-arr[i];
            if(isPrime(currentSum)){
                firstPrimeSum = currentSum;
                break;
            }
        }
        if(firstPrimeSum != -1){
            System.out.println(firstPrimeSum);
        }else{
            System.out.println("No valid prime sum found");
        }
        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 3_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Alex is a treasure hunter who collects valuable items during their quests. Each item has a specific point value, and Alex wants to maximize their score by strategically removing items one at a time.

The rule is simple: Alex removes the item with the highest point value in each step until no items are left, summing the values of the removed items to calculate the maximum score.

Help Alex to complete his task.

Input Format

The first line of input consists of an integer N, representing the size of the array.

The second line of input consists of N space-separated integers, representing the point values of the items.

Output Format

The output prints "Maximum Sum: " followed by the calculated maximum score after removing all items.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 14
7 14 21 28 35 42 49 56 63 70 77 84 91 98
Output: Maximum Sum: 735

Answer

-

Status : Skipped ***Marks : 0/10***

2. Problem Statement

Robin is a tech-savvy teenager who is diving into programming.

He is working on a project to find special elements in an array called 'leaders.' Leaders are those exceptional elements that are greater than the sum of all the elements to their right.

Assist Robin in writing this program.

Example

Input:

6

16 28 74 19 25 11

Output:

74 25 11

Explanation:

The element 16 is not greater than the sum of elements to its right (28 + 74 + 19 + 25 + 11 = 157)

The element 28 is not greater than the sum of elements to its right (74 + 19 + 25 + 11 = 129)

The element 74 is greater than the sum of elements to its right (19 + 25 + 11 = 55)

The element 19 is not greater than the sum of elements to its right (25 + 11 = 36)

The element 25 is greater than the sum of elements to its right (11)

The last element 11 is always a leader since there are no elements to its right.

So, the output is {74, 25, 11}.

Input Format

The first line of input consists of an integer N, representing the number of elements in the array.

The second line consists of N space-separated integers, representing the elements of the array.

Output Format

The output prints the special elements in the given array, that are greater than the sum of all the elements to their right.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
3 4 2 5 1
Output: 5 1

Answer

-

Status : -

Marks : 0/10

3. Problem Statement

Emma is a data analyst working with a grid-based system where each cell contains important numerical data. The grid represents spatial data, inventory records, or structured reports that require periodic updates.

Due to system updates and new requirements, Emma needs to modify the grid in the following ways:

She wants to insert either a new row or a new column at a given position.Later, she needs to delete either a row or a column from the modified matrix.

Input Format

The first line contains two integers rows and cols (the dimensions of the matrix).

The next rows lines contain cols space-separated integers representing the initial matrix.

The next line contains two integers insertType and insertIndex:

- insertType = 0 for row insertion, 1 for column insertion.
- insertIndex is the position where the new row/column should be added.

If inserting a row, the next cols integers represent the new row or If inserting a column, the next rows integers represent the new column.

The next line contains two integers deleteType and deleteIndex:

- deleteType = 0 for row deletion, 1 for column deletion.
- deleteIndex is the position to be deleted.

Output Format

The first line of output prints the string "After insertion" followed by the modified matrix with the inserted row or column.

Each row of the matrix is printed on a new line with space-separated integers.

The next line prints the string "After deletion" followed by the final matrix after the specified deletion operation.

Each row of the resulting matrix is printed on a new line with space-separated integers.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3 3
1 2 3
4 5 6
7 8 9
0 1
10 11 12
1 2
Output: After insertion
1 2 3
10 11 12
4 5 6
7 8 9
After deletion
1 2
10 11
4 5
7 8

Answer

-

Status : - ***Marks : 0/10***

4. Problem Statement:

Emma, a budding computer vision enthusiast, is working on a challenging

image processing project. She has a square image represented as a 2D matrix of integers. As part of a special filter operation, she needs to rotate the image by 90 degrees clockwise, but there's a twist — she must perform the rotation in-place, using no extra space.

This means Emma has to rotate the matrix without creating a new one. Your task is to help her implement a Java program that takes this square matrix as input and rotates it within the same structure.

Can you help Emma efficiently rotate the image so that her project can move to the next stage?

Input Format

The first line of input contains a single integer n, representing the number of rows and columns of the square matrix (i.e., the matrix is of size n x n).

The next n lines each contain n space-separated integers, representing the elements of each row of the 2D array.

Output Format

The first line of output prints "Rotated 2D Array:"

The next n lines of output print the rotated matrix.

Each line contains n space-separated integers representing a row of the rotated matrix.

Refer to the sample output for format specification.

Sample Test Case

Input: 3
1 2 3
4 5 6
7 8 9
Output: Rotated 2D Array:
7 4 1
8 5 2

9 6 3

Answer

-

Status : -

Marks : 0/10

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 4_Q1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a publishing company, editors often need to quickly analyze passages of text to check for punctuation usage. To assist them, you are asked to write a program that counts the number of specific punctuation marks in each passage.

The punctuation marks of interest are:

Commas (,)Periods (.)Question marks (?)

Input Format

The first line of input contains an integer T, representing the number of test cases (passages).

Each of the next T lines contains a single passage of text.

Output Format

For each test case, print three integers separated by spaces, representing the number of commas, periods, and question marks in the passage.

The first line of output corresponds to the first passage, the second line to the second passage, and so on.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
Hello, world. How are you?
Output: 1 1 1

Answer

```
import java.util.*;

public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int T = Integer.parseInt(sc.nextLine());
      for (int i = 0; i < T; i++) {
         String line = sc.nextLine();
         int commas = 0, periods = 0, questions = 0;
         for (char c : line.toCharArray()) {
            if (c == ',') commas++;
            else if (c == '.') periods++;
            else if (c == '?') questions++;
         }
         System.out.println(commas + " " + periods + " " + questions);
      }
   }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 4_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Anu is developing a tool for a conference registration system. Participants submit keywords related to their fields of interest. The organizer wants to sort these keywords alphabetically to generate tags for session grouping.

Write a program that accepts at least five keywords as input arguments and outputs them in sorted alphabetical order.

Input Format

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

The second line of input contains n space-separated keywords (string).

Output Format

The output prints n space separated strings representing the sorted keyword in alphabetical order.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
Blockchain Cloud AI Data Cybersecurity
Output: AI Blockchain Cloud Cybersecurity Data

Answer

```
import java.util.*;

class KeywordSorter {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        String[] keywords = new String[n];
        for (int i = 0; i < n; i++) {
            keywords[i] = sc.next();
        }
        Arrays.sort(keywords);
        for (int i = 0; i < n; i++) {
            System.out.print(keywords[i]);
            if (i < n - 1) System.out.print(" ");
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 4_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Bechan Chacha is seeking help to filter out valid mobile numbers from a list provided by his crush. He can only pick his crush's number if the list contains valid mobile numbers.

A mobile number is considered valid if:

It has exactly 10 digits.It consists only of numeric values (0–9).It does not begin with zero.

Your task is to determine whether each mobile number in the list is valid or not.

Input Format

The first line contains an integer T, representing the number of mobile numbers

to check.

The next T lines each contain a string S, representing a mobile number.

Output Format

For each mobile number S, the output print "YES" if it is valid.

Otherwise, print "NO".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
9876543210
Output: YES

Answer

```
import java.util.*;

class MobileNumberValidator {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int T = sc.nextInt();
        for (int i = 0; i < T; i++) {
            String S = sc.next();
            if (S.length() == 10 && S.matches("[1-9][0-9]{9}")) {
                System.out.println("YES");
            } else {
                System.out.println("NO");
            }
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 4_Q4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Arjun is learning how to filter words from a sentence based on grammar rules. He wants to identify the valid words in a sentence.

A word is considered valid if it satisfies all these conditions:

The word contains only alphabets (a–z, A–Z).The word length is at least 2 characters.The word should not contain digits or special characters.

Your task is to read a sentence and print all the valid words in it.

Input Format

The input contains a single line containing a sentence S.

Output Format

The output prints all the valid words separated by spaces.

If no valid word exists, print "No valid words."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Hello world1 123 ab" @#$ Hi
Output: Hello Hi

Answer

```
import java.util.Scanner;

public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String sentence = sc.nextLine();
      String[] words = sentence.split("\\s+");
      StringBuilder result = new StringBuilder();

      for (String word : words) {
         if (word.matches("[a-zA-Z]{2,}")) {
            result.append(word).append(" ");
         }
      }

      if (result.length() > 0) {
         System.out.println(result.toString().trim());
      } else {
         System.out.println("No valid words.");
      }
   }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 4_Q5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a secure banking system, customers are required to create PIN codes for accessing their accounts. The bank wants to validate these PIN codes before accepting them.

A PIN code is considered valid if:

It consists of exactly 4 digits.All characters must be numeric (0–9).It cannot contain all identical digits (e.g., 1111 is invalid).

Your task is to determine whether each PIN code in the list is valid or not.

Input Format

The first line of input contains an integer T, representing the number of PIN codes to check.

The next T lines each contain a string S, representing a PIN code.

Output Format

For each PIN code S, the output print "YES" if it is valid.

Otherwise, the output print "NO".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1234
Output: YES

Answer

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int T = Integer.parseInt(sc.nextLine());
        for (int i = 0; i < T; i++) {
            String pin = sc.nextLine();
            if (pin.matches("\\d{4}") && !pin.chars().allMatch(ch -> ch ==
pin.charAt(0))) {
                System.out.println("YES");
            } else {
                System.out.println("NO");
            }
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 4_PAH

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

At a digital library, the system needs to analyze passages to identify the frequency of vowels, since they are key for linguistic research. You are asked to write a program that counts the number of vowels in each passage of text.

The vowels of interest are:

a, e, i, o, u (both uppercase and lowercase).

Input Format

The first line of input contains an integer T, representing the number of test cases (passages).

Each of the next T lines contains a single passage of text.

Output Format

For each test case, print a single integer representing the total number of vowels in the passage.

The first line of output corresponds to the first passage, the second line to the second passage, and so on.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
Hello World
Output: 3

Answer

```
import java.util.Scanner;

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int T = Integer.parseInt(sc.nextLine());
        for (int i = 0; i < T; i++) {
            String passage = sc.nextLine();
            int count = 0;
            for (char c : passage.toCharArray()) {
                if ("aeiouAEIOU".indexOf(c) != -1) {
                    count++;
                }
            }
            System.out.println(count);
        }
    }
}
```

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

2. Problem Statement

Ravi is analyzing text messages for his research on typing patterns. He wants to count the number of uppercase letters, lowercase letters, and digits in a sentence to understand typing trends.

Your task is to help Ravi by writing a program that takes a sentence and prints the count of uppercase letters, lowercase letters, and digits.

Input Format

The input contains a single line containing a sentence (string).

Output Format

The output prints three integers separated by spaces:

- Number of uppercase letters
- Number of lowercase letters
- Number of digits

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Hello World 123
Output: 2 8 3

Answer

```
import java.util.Scanner;

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String sentence = sc.nextLine();
        int upper = 0, lower = 0, digits = 0;
        for (char c : sentence.toCharArray()) {
            if (Character.isUpperCase(c)) upper++;
            else if (Character.isLowerCase(c)) lower++;
            else if (Character.isDigit(c)) digits++;
        }
        System.out.println(upper + " " + lower + " " + digits);
```

```
    }
}
```

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

3. Problem Statement

Sana is analyzing text for a secret code. She wants to find all words in a sentence that start and end with the same letter. These words are considered "special words" for her analysis.

Your task is to write a program that extracts and prints all words that start and end with the same letter (case-insensitive).

If no such word exists, print "No special words found".

Input Format

The input contains a single line containing a sentence with multiple words.

Output Format

The output prints all words that start and end with the same letter separated by a space.

If no word satisfies the condition, print "No special words found".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Anna went to the civic center
Output: Anna civic

Answer

```
import java.util.Scanner;

class Main {
    public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
        String[] words = sc.nextLine().split("\\s+");
        StringBuilder sb = new StringBuilder();
        for (String word : words) {
            if (word.length() > 0) {
                char start = Character.toLowerCase(word.charAt(0));
                char end = Character.toLowerCase(word.charAt(word.length() - 1));
                if (start == end) {
                    sb.append(word).append(" ");
                }
            }
        }
        if (sb.length() > 0) {
            System.out.println(sb.toString().trim());
        } else {
            System.out.println("No special words found");
        }
    }
}
```

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

4. Problem Statement

Riya is preparing a puzzle game for her friends. She wants to include a feature that highlights special words in a sentence — specifically, palindromic words (words that read the same forward and backward).

Your task is to help Riya by writing a program that extracts all palindrome words from the given sentence. If there are no palindromes, print "No palindromes found".

Input Format

The input contains a single string S representing a sentence.

Output Format

The output prints all palindromic words separated by a space.

If no palindrome exists, print "No palindromes found".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: madam went to school

Output: madam

Answer

```
import java.util.Scanner;

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String[] words = sc.nextLine().split("\\s+");
        StringBuilder sb = new StringBuilder();
        for (String word : words) {
            String lower = word.toLowerCase();
            if (isPalindrome(lower)) {
                sb.append(word).append(" ");
            }
        }
        if (sb.length() > 0) {
            System.out.println(sb.toString().trim());
        } else {
            System.out.println("No palindromes found");
        }
    }

    static boolean isPalindrome(String s) {
        int left = 0, right = s.length() - 1;
        while (left < right) {
            if (s.charAt(left) != s.charAt(right)) return false;
            left++;
            right--;
        }
        return true;
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 4_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Neha is analyzing text messages to identify words that have repeated characters. A word is considered “repetitive” if any character appears more than once in that word.

Your task is to write a program that extracts all words that contain repeated characters from a given sentence.

If no such word exists, print "No repetitive words found".

Input Format

The input contains a single line containing a sentence with multiple words.

Output Format

The output prints all words that contain repeated characters separated by a space.

If no word contains repeated characters, print "No repetitive words found".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: letter balloon apple tree

Output: letter balloon apple tree

Answer

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

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String[] words = sc.nextLine().split("\\s+");
        StringBuilder sb = new StringBuilder();

        for (String word : words) {
            if (hasRepeatedChars(word)) {
                sb.append(word).append(" ");
            }
        }
        if (sb.length() > 0) {
            System.out.println(sb.toString().trim());
        } else {
            System.out.println("No repetitive words found");
        }
    }

    static boolean hasRepeatedChars(String word) {
        HashSet<Character> set = new HashSet<>();
        for (char c : word.toCharArray()) {
            if (set.contains(c)) return true;
            set.add(c);
        }
```

```
        return false;
    }
}
```

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

2. Problem Statement

Meera is practicing her English vocabulary. She wants to focus on words that have more vowels in them, as they help improve her pronunciation. She decides to extract only those words from a sentence that contain at least two vowels.

Your task is to help Meera by writing a program that finds such words from the given sentence.

Input Format

The input contains a string representing the sentence.

Output Format

The output prints all the words that contain at least two vowels, separated by a space.

If no such word exists, print "No words with two vowels".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: This is an example sentence
Output: example sentence

Answer

```
import java.util.Scanner;

class Main {
    public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
        String[] words = sc.nextLine().split("\\s+");
        StringBuilder sb = new StringBuilder();
        for (String word : words) {
            if (countVowels(word) >= 2) {
                sb.append(word).append(" ");
            }
        }
        if (sb.length() > 0) {
            System.out.println(sb.toString().trim());
        } else {
            System.out.println("No words with two vowels");
        }
    }

    static int countVowels(String word) {
        int count = 0;
        for (char c : word.toLowerCase().toCharArray()) {
            if ("aeiou".indexOf(c) != -1) count++;
        }
        return count;
    }
}
```

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

3. Problem Statement

In a college, students are required to create unique usernames for accessing the digital library.

The librarian needs your help to verify whether the usernames entered by students are valid.

A username is considered valid if:

It contains only letters (a–z, A–Z) and digits (0–9).Its length is between 5 and 15 characters (inclusive).It must start with a letter (not a digit).

Your task is to determine whether each username in the list is valid or not.

Input Format

The first line of input contains an integer T, representing the number of usernames to check.

The next T lines each contain a string S, representing a username.

Output Format

For each username S, the output print "YES" if it is valid.

Otherwise, the output print "NO".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
Alice123
Output: YES

Answer

```
import java.util.Scanner;

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int T = Integer.parseInt(sc.nextLine());
        for (int i = 0; i < T; i++) {
            String username = sc.nextLine();
            if (username.matches("^[a-zA-Z][a-zA-Z0-9]{4,14}$")) {
                System.out.println("YES");
            } else {
                System.out.println("NO");
            }
        }
    }
}
```

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

4. Problem Statement

A bookstore wants to analyze the titles of books to determine their longest word in each title. This helps in designing banners and covers.

Your task is to write a program that, given a sentence (book title), finds and prints the longest word. If multiple words have the same maximum length, print the first one.

Input Format

The input contais a single line containing a sentence representing the book title.

Output Format

The output prints a string representing the longest word in the sentence (book title).

Refer to the sample output for formatting specifications.

Sample Test Case

Input: The Chronicles of Narnia

Output: Chronicles

Answer

```
import java.util.Scanner;

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Read the entire line as the book title
        String title = sc.nextLine();
        // Split the title into words using whitespace as delimiter
        String[] words = title.split("\\s+");

        String longestWord = "";
        // Iterate over each word to find the longest one
        for (String word : words) {
            if (word.length() > longestWord.length()) {
```

```
                longestWord = word;
            }
        }
        // Print the longest word found
        System.out.println(longestWord);
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 5_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are working as a developer for CityBank, which wants to build a basic account management system.

Each customer at the bank has:

An Account Number (integer)A Customer Name (string)An Initial Balance (double)

The bank allows two types of transactions:

Deposit – increases the balance.Withdrawal – decreases the balance only if enough funds are available.

If the withdrawal amount is greater than the balance, the withdrawal should not happen, and the balance should remain the same.

You are required to implement this system using:

A class with attributes for account details.A constructor to initialize account details.Setter methods to update details if needed.Getter methods to retrieve details.Objects of the class to represent customers.

Finally, display each customer’s account details after all transactions.

Input Format

The first line of input contains an integer N, representing the number of customers.

For each customer:

- The next line contains the account number (integer).
- The following line contains the customer name (string).
- The next line contains the initial balance (double).
- The next line contains the deposit amount (double).
- The next line contains the withdrawal amount (double).

Output Format

For each customer, print the details in the following format:

1. Account Number: <account_number>
2. Customer Name: <customer_name>
3. Final Balance: <final_balance> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1234
Rahul Sharma
5000
2000
3000
Output: Account Number: 1234
Customer Name: Rahul Sharma

Final Balance: 4000.0

Answer

```
import java.util.Scanner;
class Account {
    private int accountNumber;
    private String customerName;
    private double balance;
    public Account(int accountNumber, String customerName, double balance) {
        this.accountNumber = accountNumber;
        this.customerName = customerName;
        this.balance = balance;
    }
    public void setAccountNumber(int accountNumber) {
        this.accountNumber = accountNumber;
    }
    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }
    public void setBalance(double balance) {
        this.balance = balance;
    }
    public int getAccountNumber() {
        return accountNumber;
    }
    public String getCustomerName() {
        return customerName;
    }
    public double getBalance() {
        return balance;
    }
    public void deposit(double amount) {
        balance += amount;
    }
    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
        }
    }
}

public class Main {
```

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

        for (int i = 0; i < N; i++) {
            int accNo = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            double initialBalance = Double.parseDouble(sc.nextLine());
            double depositAmount = Double.parseDouble(sc.nextLine());
            double withdrawalAmount = Double.parseDouble(sc.nextLine());

            Account acc = new Account(accNo, name, initialBalance);
            acc.deposit(depositAmount);
            acc.withdraw(withdrawalAmount);

            System.out.printf("Account Number: %d%n", acc.getAccountNumber());
            System.out.printf("Customer Name: %s%n", acc.getCustomerName());
            System.out.printf("Final Balance: %.1f%n", acc.getBalance());
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 5_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Neha is working as a developer for CityElectricity Board, which wants to build a household electricity billing system.

Each customer's electricity account has:

A Customer ID (integer)A Customer Name (string)Units Consumed (double)

The electricity bill is calculated based on these rules:

For the first 100 units 5 units charge per unitFor the next 100 units (101–200) 7 units charge per unitFor units above 200 10 units charge per unitIf the total bill exceeds 2000 units, a 5% discount is applied on the final bill.

Neha has been asked to implement this system using:

A class with attributes for customer details.A constructor to initialize customer details.Setter methods to update details if needed.Getter methods to retrieve details.Objects of the class to represent customers.

Finally, display each customer's details and final bill amount.

Input Format

The first line of input contains an integer N, representing the number of customers.

For each customer:

- The next line contains the Customer ID (integer).
- The following line contains the Customer Name (string).
- The next line contains the Units Consumed (double).

Output Format

For each customer, print the details in the following format:

Customer ID: <customer_id>

Customer Name: <customer_name>

Final Bill: <final_bill> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001
Ravi Kumar
80
Output: Customer ID: 1001
Customer Name: Ravi Kumar
Final Bill: 400.0

Answer

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

```
class Customer {
    private int customerId;
    private String customerName;
    private double unitsConsumed;

    public Customer(int customerId, String customerName, double
unitsConsumed) {
        this.customerId = customerId;
        this.customerName = customerName;
        this.unitsConsumed = unitsConsumed;
    }

    public int getCustomerId() {
        return customerId;
    }

    public String getCustomerName() {
        return customerName;
    }

    public double getUnitsConsumed() {
        return unitsConsumed;
    }

    public void setCustomerId(int customerId) {
        this.customerId = customerId;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public void setUnitsConsumed(double unitsConsumed) {
        this.unitsConsumed = unitsConsumed;
    }

    public double calculateFinalBill() {
        double bill = 0;
        if (unitsConsumed <= 100) {
            bill = unitsConsumed * 5;
        } else if (unitsConsumed <= 200) {
```

```
            bill = 100 * 5 + (unitsConsumed - 100) * 7;
        } else {
            bill = 100 * 5 + 100 * 7 + (unitsConsumed - 200) * 10;
        }
        if (bill > 2000) {
            bill *= 0.95;
        }
        return bill;
    }
}

class ElectricityBillingSystem {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        List<Customer> customers = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            int customerId = Integer.parseInt(sc.nextLine());
            String customerName = sc.nextLine();
            double unitsConsumed = Double.parseDouble(sc.nextLine());
            customers.add(new Customer(customerId, customerName,
unitsConsumed));
        }

        for (Customer c : customers) {
            System.out.println("Customer ID: " + c.getCustomerId());
            System.out.println("Customer Name: " + c.getCustomerName());
            System.out.printf("Final Bill: %.1f\n", c.calculateFinalBill());
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 5_Q4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are working as a developer for CityCab, a taxi service company that wants to build a ride fare management system.

Each customer booking has:

A Booking ID (integer)A Customer Name (string)A Distance Travelled in km (double)

The fare calculation rules are:

Base Fare = 50 units (flat charge for every ride).Per km charge = 10 units/km.If the distance is greater than 20 km, a 10% discount is applied on the total fare.

You are required to implement this system using:

A class with attributes for booking details.A constructor to initialize booking details.Setter methods to update details if needed.Getter methods to retrieve details.Objects of the class to represent customer rides.

Finally, display each booking's details and final fare.

Input Format

The first line of input contains an integer N, representing the number of bookings.

For each booking:

- The next line contains the booking ID (integer).
- The following line contains the customer's name (string).
- The next line contains the distance travelled (double).

Output Format

For each booking, print the details in the following format:

1. Booking ID: <booking_id>
2. Customer Name: <customer_name>
3. Final Fare: <final_fare> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1234
Rahul Sharma
15

Output: Booking ID: 1234
Customer Name: Rahul Sharma
Final Fare: 200.0

Answer

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

```
class Booking {
    private int bookingId;
    private String customerName;
    private double distance;
    private double fare;

    public Booking(int bookingId, String customerName, double distance) {
        this.bookingId = bookingId;
        this.customerName = customerName;
        this.distance = distance;
        calculateFare();
    }

    public void setBookingId(int bookingId) {
        this.bookingId = bookingId;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public void setDistance(double distance) {
        this.distance = distance;
        calculateFare();
    }

    public int getBookingId() {
        return bookingId;
    }

    public String getCustomerName() {
        return customerName;
    }

    public double getDistance() {
        return distance;
    }

    public double getFare() {
        return fare;
    }
```

```
    private void calculateFare() {
        fare = 50 + distance * 10;
        if (distance > 20) {
            fare = fare - (fare * 0.1);
        }
    }
}

class CityCabApp {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        for (int i = 0; i < n; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            double distance = Double.parseDouble(sc.nextLine());
            Booking booking = new Booking(id, name, distance);
            System.out.println("Booking ID: " + booking.getBookingId());
            System.out.println("Customer Name: " + booking.getCustomerName());
            System.out.printf("Final Fare: %.1f\n", booking.getFare());
        }
        sc.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 5_Q5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ram is working as a developer for BrightEdu Coaching Center, which wants to build a student fee management system.

Each student's enrollment has:

An Enrollment ID (integer)A Student Name (string)The Number of Subjects (integer)

The fee calculation rules are:

Registration Fee = 1000 units (flat for every student).Per Subject Fee = 800 units.If the student enrolls in more than 5 subjects, a 20% scholarship (discount) is applied on the total fee.

Ram has been asked to implement this system using:

A class with attributes for student details.A constructor to initialize student details.Setter methods to update details if needed.Getter methods to retrieve details.Objects of the class to represent student enrollments.

Finally, display each student's details and final fee.

Input Format

The first line of input contains an integer N, representing the number of students.

For each student:

- The next line contains the Enrollment ID (integer).
- The following line contains the student's name (string).
- The next line contains the Number of subjects (integer).

Output Format

For each student, print the details in the following format:

- Enrollment ID: <enrollment_id>
- Student Name: <student_name>
- Final Fee: <final_fee> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1234
Ravi Kumar
3
Output: Enrollment ID: 1234
Student Name: Ravi Kumar
Final Fee: 3400.0

Answer

```
import java.util.*;

class Student {
    private int enrollmentId;
```

```
    private String studentName;
    private int numberOfSubjects;

    public Student(int enrollmentId, String studentName, int numberOfSubjects) {
        this.enrollmentId = enrollmentId;
        this.studentName = studentName;
        this.numberOfSubjects = numberOfSubjects;
    }

    public int getEnrollmentId() {
        return enrollmentId;
    }

    public String getStudentName() {
        return studentName;
    }

    public int getNumberOfSubjects() {
        return numberOfSubjects;
    }

    public void setEnrollmentId(int enrollmentId) {
        this.enrollmentId = enrollmentId;
    }

    public void setStudentName(String studentName) {
        this.studentName = studentName;
    }

    public void setNumberOfSubjects(int numberOfSubjects) {
        this.numberOfSubjects = numberOfSubjects;
    }

    public double calculateFinalFee() {
        double registrationFee = 1000;
        double subjectFee = numberOfSubjects * 800;
        double totalFee = registrationFee + subjectFee;
        if (numberOfSubjects > 5) {
            totalFee *= 0.8;
        }
        return totalFee;
    }
```

```
}

class BrightEduFeeSystem {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        List<Student> students = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            int enrollmentId = Integer.parseInt(sc.nextLine());
            String studentName = sc.nextLine();
            int numberOfSubjects = Integer.parseInt(sc.nextLine());
            students.add(new Student(enrollmentId, studentName,
numberOfSubjects));
        }

        for (Student s : students) {
            System.out.println("Enrollment ID: " + s.getEnrollmentId());
            System.out.println("Student Name: " + s.getStudentName());
            System.out.printf("Final Fee: %.1f\n", s.calculateFinalFee());
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 5_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Neha is working as a developer for CityQuiz Platform, which wants to build a system to calculate quiz scores and identify top scorers among participants.

Each participant's record has:

Participant ID (integer)Participant Name (string)An array of scores in 5 quiz rounds (integers, each between 0 and 100)

The system must calculate:

Total Score = sum of scores in all 5 rounds.Average Score = Total Score ÷ 5.If a participant scores above 80 in all rounds, a bonus of 10 points is added to the total score.Identify the Top Scorer among all participants. If

two participants have the same total score, the one with the lower Participant ID is considered the top scorer.

Neha has been asked to implement this system using:

A class with attributes for participant details.A constructor to initialize participant details.Getter and setter methods to retrieve or update participant details.A method to calculate total score and average score (including bonus if applicable).Objects of the class to represent participants.

Finally, display each participant's details and announce the Top Scorer.

Input Format

The first line of input contains an integer N, representing the number of participants.

For each participant:

- Next line: Participant ID (integer)
- Next line: Participant Name (string)
- Next line: 5 integers separated by spaces (scores for 5 quiz rounds)

Output Format

For each participant:

- Participant ID: <participant_id>
- Participant Name: <participant_name>
- Total Score: <total_score>
- Average Score: <average_score>

Finally, print "Top Scorer: <participant_name> with <total_score> points"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001
Ravi Kumar
85 90 88 92 87

Output: Participant ID: 1001
Participant Name: Ravi Kumar
Total Score: 452
Average Score: 90
Top Scorer: Ravi Kumar with 452 points

Answer

```
import java.util.*;

class Participant {
    private int participantId;
    private String participantName;
    private int[] scores;
    private int totalScore;
    private int averageScore;

    public Participant(int participantId, String participantName, int[] scores) {
        this.participantId = participantId;
        this.participantName = participantName;
        this.scores = scores;
    }

    public int getParticipantId() {
        return participantId;
    }

    public String getParticipantName() {
        return participantName;
    }

    public int getTotalScore() {
        return totalScore;
    }

    public int getAverageScore() {
        return averageScore;
    }
```

```
    public void calculateScores() {
        totalScore = 0;
        boolean bonusEligible = true;
        for (int score : scores) {
            totalScore += score;
            if (score <= 80) {
                bonusEligible = false;
            }
        }
        if (bonusEligible) {
            totalScore += 10;
        }
        averageScore = totalScore / 5;
    }
}

class CityQuiz {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        List<Participant> participants = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            String[] scoreStr = sc.nextLine().split(" ");
            int[] scores = new int[5];
            for (int j = 0; j < 5; j++) {
                scores[j] = Integer.parseInt(scoreStr[j]);
            }
            Participant p = new Participant(id, name, scores);
            p.calculateScores();
            participants.add(p);
        }

        Participant topScorer = participants.get(0);
        for (Participant p : participants) {
            System.out.println("Participant ID: " + p.getParticipantId());
            System.out.println("Participant Name: " + p.getParticipantName());
            System.out.println("Total Score: " + p.getTotalScore());
            System.out.println("Average Score: " + p.getAverageScore());
            if (p.getTotalScore() > topScorer.getTotalScore() ||
```

```
            (p.getTotalScore() == topScorer.getTotalScore() &&
             p.getParticipantId() < topScorer.getParticipantId())) {
            topScorer = p;
        }
    }

    System.out.println("Top Scorer: " + topScorer.getParticipantName() +
                       " with " + topScorer.getTotalScore() + " points");
  }
}
```

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

2. Problem Statement

Neha is working as a developer for CityMovie Theatre, which wants to build a system to calculate total ticket cost for movie-goers based on the number of tickets and type of seats booked.

Each customer’s booking has:

Booking ID (integer)Customer Name (string)Number of Tickets (integer)Seat Type (string: "Standard", "Premium", "VIP")

The ticket prices are:

Standard – 250 units per ticketPremium – 400 units per ticketVIP – 600 units per ticket

The calculation rules:

Total Amount = Number of Tickets × Seat Price

If a customer books more than 4 tickets, they get a 10% discount on the total amount.

If the booking is for VIP seats and the total amount exceeds 3000 units, a 5% luxury tax is added after any discount.

Neha has been asked to implement this system using:

A class with attributes for booking details.A constructor to initialize booking details.Getter and Setter methods to retrieve and update booking details if required.A method to calculate the final ticket cost.Objects of the class to represent bookings.

Finally, display each customer's details and final ticket amount.

Input Format

The first line contains an integer N, representing the number of bookings.

For each booking:

- The next line contains the Booking ID (integer).
- The next line contains the Customer Name (string).
- The next line contains Number of Tickets (integer).
- The next line contains Seat Type ("Standard", "Premium", or "VIP").

Output Format

For each booking, print:

- Booking ID: <booking_id>
- Customer Name: <customer_name>
- Final Ticket Amount: <final_amount> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001
Ravi Kumar
3
Standard
Output: Booking ID: 1001
Customer Name: Ravi Kumar
Final Ticket Amount: 750.0

Answer

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

```
class Booking {
    private int bookingId;
    private String customerName;
    private int numberOfTickets;
    private String seatType;

    public Booking(int bookingId, String customerName, int numberOfTickets,
String seatType) {
        this.bookingId = bookingId;
        this.customerName = customerName;
        this.numberOfTickets = numberOfTickets;
        this.seatType = seatType;
    }

    public int getBookingId() {
        return bookingId;
    }

    public String getCustomerName() {
        return customerName;
    }

    public int getNumberOfTickets() {
        return numberOfTickets;
    }

    public String getSeatType() {
        return seatType;
    }

    public void setBookingId(int bookingId) {
        this.bookingId = bookingId;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public void setNumberOfTickets(int numberOfTickets) {
        this.numberOfTickets = numberOfTickets;
    }
```

```
    public void setSeatType(String seatType) {
        this.seatType = seatType;
    }

    public double calculateFinalAmount() {
        double pricePerTicket = 0;
        if (seatType.equalsIgnoreCase("Standard")) pricePerTicket = 250;
        else if (seatType.equalsIgnoreCase("Premium")) pricePerTicket = 400;
        else if (seatType.equalsIgnoreCase("VIP")) pricePerTicket = 600;

        double totalAmount = numberOfTickets * pricePerTicket;

        if (numberOfTickets > 4) totalAmount *= 0.9;

        if (seatType.equalsIgnoreCase("VIP") && totalAmount > 3000) totalAmount
*= 1.05;

        return totalAmount;
    }
}

class CityMovieTheatre {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        List<Booking> bookings = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            int bookingId = Integer.parseInt(sc.nextLine());
            String customerName = sc.nextLine();
            int numberOfTickets = Integer.parseInt(sc.nextLine());
            String seatType = sc.nextLine();
            bookings.add(new Booking(bookingId, customerName, numberOfTickets,
seatType));
        }

        for (Booking b : bookings) {
            System.out.println("Booking ID: " + b.getBookingId());
            System.out.println("Customer Name: " + b.getCustomerName());
            System.out.printf("Final Ticket Amount: %.1f\n",
b.calculateFinalAmount());
```

```
    }
  }
}
```

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

3. Problem Statement

Ravi is working as a developer for SecureLogin Systems, which wants to build a system to evaluate the strength of user passwords.

Each user record has:

User ID (integer)User Name (string)Password (string)

The system must calculate whether a password is strong or weak.

A password is considered strong if it meets all of the following conditions:

At least 8 characters long.Contains at least one uppercase letter.Contains at least one lowercase letter.Contains at least one digit.Contains at least one special character (from !@#$%^&*).

Ravi has been asked to implement this system using:

A class with attributes for user details.A constructor to initialize user details.Getter and setter methods to retrieve or update user details.A method to check whether the password is strong.Objects of the class to represent users.

Finally, display each user's details and indicate whether their password is Strong or Weak.

Input Format

The first line contains an integer N, representing the number of users.

For each user:

The next line contains the User ID (integer).

The next line contains the User Name (string).

The next line contains the Password (string).

Output Format

For each user, print the details in the following format:

User ID: <user_id>

User Name: <user_name>

Password: <password>

Password Strength: <Strong/Weak>

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001
Ravi Kumar
Abc@1234
Output: User ID: 1001
User Name: Ravi Kumar
Password: Abc@1234
Password Strength: Strong

Answer

```
import java.util.*;

class User {
    private int userId;
    private String userName;
    private String password;

    public User(int userId, String userName, String password) {
        this.userId = userId;
        this.userName = userName;
        this.password = password;
```

```
    }

    public int getUserId() {
        return userId;
    }

    public String getUserName() {
        return userName;
    }

    public String getPassword() {
        return password;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String checkPasswordStrength() {
        if (password.length() < 8) return "Weak";
        if (!password.matches(".*[A-Z].*")) return "Weak";
        if (!password.matches(".*[a-z].*")) return "Weak";
        if (!password.matches(".*\\d.*")) return "Weak";
        if (!password.matches(".*[!@#$%^&*].*")) return "Weak";
        return "Strong";
    }
}

class SecureLoginSystem {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        List<User> users = new ArrayList<>();
```

```
        for (int i = 0; i < n; i++) {
            int userId = Integer.parseInt(sc.nextLine());
            String userName = sc.nextLine();
            String password = sc.nextLine();
            users.add(new User(userId, userName, password));
        }

        for (User u : users) {
            System.out.println("User ID: " + u.getUserId());
            System.out.println("User Name: " + u.getUserName());
            System.out.println("Password: " + u.getPassword());
            System.out.println("Password Strength: " + u.checkPasswordStrength());
        }
    }
}
```

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

4. Problem Statement

Each customer at the bank has an Account Number, Customer Name, and an Initial Balance. The bank allows two types of transactions:

Deposit – Increases the balance.Withdrawal – Decreases the balance, but only if enough funds are available. If the withdrawal amount exceeds the available balance, the transaction should be skipped, and the balance should remain unchanged.

You are required to implement this banking system by:

Creating a class with the necessary attributes to store account details.

Using a constructor to initialize the account details when a new account is created.Providing setter methods to update the details if required.Providing getter methods to retrieve account details.Creating objects of this class to represent different customers, where each customer can perform deposits and withdrawals.

Instructions:

Implement the class to store account details.Implement the logic for

performing deposit and withdrawal transactions.Ensure that withdrawals don't exceed the available balance.After performing the transactions, print the account number, customer name, and final balance.

Input Format

The first line of input contains an integer N, representing the number of customers.

For each customer:

- The next line contains the account number (integer).
- The following line contains the customer name (string).
- The next line contains the initial balance (double).
- The next line contains the deposit amount (double).
- The next line contains the withdrawal amount (double).

Output Format

For each customer, print the details in the following format:

1. Account Number: <account_number>
2. Customer Name: <customer_name>
3. Final Balance: <final_balance> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1234
Rahul Sharma
5000
2000
3000

Output: Account Number: 1234
Customer Name: Rahul Sharma
Final Balance: 4000.0

Answer

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

```
class BankAccount {
    private int accountNumber;
    private String customerName;
    private double balance;

    public BankAccount(int accountNumber, String customerName, double
balance) {
        this.accountNumber = accountNumber;
        this.customerName = customerName;
        this.balance = balance;
    }

    public int getAccountNumber() {
        return accountNumber;
    }

    public String getCustomerName() {
        return customerName;
    }

    public double getBalance() {
        return balance;
    }

    public void setAccountNumber(int accountNumber) {
        this.accountNumber = accountNumber;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }

    public void deposit(double amount) {
        balance += amount;
    }

    public void withdraw(double amount) {
```

```
        if (amount <= balance) {
            balance -= amount;
        }
    }
}

class BankingSystem {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        List<BankAccount> accounts = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            int accountNumber = Integer.parseInt(sc.nextLine());
            String customerName = sc.nextLine();
            double initialBalance = Double.parseDouble(sc.nextLine());
            double depositAmount = Double.parseDouble(sc.nextLine());
            double withdrawalAmount = Double.parseDouble(sc.nextLine());

            BankAccount account = new BankAccount(accountNumber,
customerName, initialBalance);
            account.deposit(depositAmount);
            account.withdraw(withdrawalAmount);
            accounts.add(account);
        }

        for (BankAccount acc : accounts) {
            System.out.println("Account Number: " + acc.getAccountNumber());
            System.out.println("Customer Name: " + acc.getCustomerName());
            System.out.printf("Final Balance: %.1f\n", acc.getBalance());
        }
    }
}
```

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

5. Problem Statement

Anjali is working as a developer for CityFitness Gym, which wants to build a system to calculate monthly membership fees for gym members based on the type of membership and the number of personal training sessions

booked.

Each member’s record has:

Member ID (integer)Member Name (string)Membership Type (string: "Basic", "Premium", "Elite")Number of Personal Training Sessions (integer)

The monthly fees are:

Basic – 1000 unitsPremium – 1500 unitsElite – 2000 units

The cost of personal training sessions is 500 units per session.

The calculation rules:

Total Amount = Membership Fee + (Number of Personal Training Sessions × 500)If the number of sessions is more than 5, a 10% discount is applied on the total amount.If the member has Elite membership and the total amount exceeds 4000, an additional 5% service tax is added after discount.

Anjali has been asked to implement this system using:

A class with attributes for member details.A constructor to initialize member details.Getter and Setter methods to retrieve and update member details if required.A method to calculate the final monthly fee.Objects of the class to represent members.

Finally, display each member’s details and the final monthly fee.

Input Format

The first line contains an integer N, representing the number of members.

For each member:

- Next line contains Member ID (integer)
- Next line contains Member Name (string)
- Next line contains Membership Type ("Basic", "Premium", "Elite")
- Next line contains Number of Personal Training Sessions (integer)

Output Format

For each member, print:

- Member ID: <member_id>
- Member Name: <member_name>
- Final Monthly Fee: <final_fee> (The final fee must be rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001
Ravi Kumar
Basic
3

Output: Member ID: 1001
Member Name: Ravi Kumar
Final Monthly Fee: 2500.0

Answer

```
import java.util.*;

class Member {
    private int memberId;
    private String memberName;
    private String membershipType;
    private int trainingSessions;

    public Member(int memberId, String memberName, String membershipType,
int trainingSessions) {
        this.memberId = memberId;
        this.memberName = memberName;
        this.membershipType = membershipType;
        this.trainingSessions = trainingSessions;
    }

    public int getMemberId() {
        return memberId;
    }
```

```
    public String getMemberName() {
        return memberName;
    }

    public String getMembershipType() {
        return membershipType;
    }

    public int getTrainingSessions() {
        return trainingSessions;
    }

    public void setMemberId(int memberId) {
        this.memberId = memberId;
    }

    public void setMemberName(String memberName) {
        this.memberName = memberName;
    }

    public void setMembershipType(String membershipType) {
        this.membershipType = membershipType;
    }

    public void setTrainingSessions(int trainingSessions) {
        this.trainingSessions = trainingSessions;
    }

    public double calculateFinalFee() {
        double membershipFee = 0;
        if (membershipType.equalsIgnoreCase("Basic")) membershipFee = 1000;
        else if (membershipType.equalsIgnoreCase("Premium")) membershipFee =
1500;
        else if (membershipType.equalsIgnoreCase("Elite")) membershipFee = 2000;

        double sessionCost = trainingSessions * 500;
        double total = membershipFee + sessionCost;

        if (trainingSessions > 5) total *= 0.9;

        if (membershipType.equalsIgnoreCase("Elite") && total > 4000) total *= 1.05;
```

```
        return total;
    }
}

class CityFitnessGym {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        List<Member> members = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            int memberId = Integer.parseInt(sc.nextLine());
            String memberName = sc.nextLine();
            String membershipType = sc.nextLine();
            int trainingSessions = Integer.parseInt(sc.nextLine());
            members.add(new Member(memberId, memberName, membershipType,
trainingSessions));
        }

        for (Member m : members) {
            System.out.println("Member ID: " + m.getMemberId());
            System.out.println("Member Name: " + m.getMemberName());
            System.out.printf("Final Monthly Fee: %.1f\n", m.calculateFinalFee());
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 5_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Arjun is working as a developer for CityWater Supply Board, which wants to build a household water billing system.

Each household's water account has:

A Customer ID (integer)A Customer Name (string)Liters Consumed (double)

The water bill is calculated based on these rules:

For the first 500 liters 2 per literFor the next 500 liters (501–1000) 3 per literFor liters above 1000 5 per literIf the total bill exceeds 3000, a 10% discount is applied on the final bill.

Arjun has been asked to implement this system using:

A class with attributes for customer details.A constructor to initialize customer details.Setter methods to update details if needed.Getter methods to retrieve details.Objects of the class to represent customers.

Finally, display each customer's details and final bill amount.

Input Format

The first line of input contains an integer N, representing the number of customers.

For each customer:

- The next line contains the Customer ID (integer).
- The following line contains the Customer Name (string).
- The next line contains the Liters Consumed (double).

Output Format

For each customer, print the details in the following format:

Customer ID: <customer_id>

Customer Name: <customer_name>

Final Bill: <final_bill> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001
Ravi Kumar
300
Output: Customer ID: 1001
Customer Name: Ravi Kumar
Final Bill: 600.0

Answer

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

```
class Customer {
    private int customerId;
    private String customerName;
    private double litersConsumed;
    private double finalBill;

    public Customer(int customerId, String customerName, double
litersConsumed) {
        this.customerId = customerId;
        this.customerName = customerName;
        this.litersConsumed = litersConsumed;
        calculateFinalBill();
    }

    private void calculateFinalBill() {
        double bill = 0;
        double liters = litersConsumed;

        if (liters > 1000) {
            bill += (liters - 1000) * 5;
            liters = 1000;
        }
        if (liters > 500) {
            bill += (liters - 500) * 3;
            liters = 500;
        }
        bill += liters * 2;

        if (bill > 3000) {
            bill = bill - (bill * 0.10);
        }

        this.finalBill = bill;
    }

    public int getCustomerId() {
        return customerId;
    }

    public String getCustomerName() {
        return customerName;
```

```
    }

    public double getFinalBill() {
        return finalBill;
    }

    public void display() {
        System.out.printf("Customer ID: %d\n", customerId);
        System.out.printf("Customer Name: %s\n", customerName);
        System.out.printf("Final Bill: %.1f\n", finalBill);
    }
}

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

        for (int i = 0; i < n; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            double liters = Double.parseDouble(sc.nextLine());

            Customer customer = new Customer(id, name, liters);
            customer.display();
        }
    }
}
```

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

2. Problem Statement

Meera is working as a developer for CityGas Supply Board, which wants to build a household gas billing system.

Each household’s gas account has:

A Customer ID (integer)A Customer Name (string)Units Consumed in cubic meters (double)

The gas bill is calculated based on these rules:

For the first 50 units 4 per unitFor the next 100 units (51–150) 6 per unitFor units above 150 8 per unitIf the total bill exceeds 2000, a 15% discount is applied on the final bill.

Meera has been asked to implement this system using:

A class with attributes for customer details.A constructor to initialize customer details.Setter methods to update details if needed.Getter methods to retrieve details.Objects of the class to represent customers.

Finally, display each customer's details and final bill amount.

Input Format

The first line of input contains an integer N, representing the number of customers.

For each customer:

- The next line contains the Customer ID (integer).
- The following line contains the Customer Name (string).
- The next line contains the Units Consumed (double).

Output Format

For each customer, print the details in the following format:

Customer ID: <customer_id>

Customer Name: <customer_name>

Final Bill: <final_bill> (The final bill must be rounded to one decimal place.)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001

Ravi Kumar
30
Output: Customer ID: 1001
Customer Name: Ravi Kumar
Final Bill: 120.0

Answer

```
import java.util.Scanner;

class Customer {
    private int customerId;
    private String customerName;
    private double unitsConsumed;

    public Customer(int customerId, String customerName, double
unitsConsumed) {
        this.customerId = customerId;
        this.customerName = customerName;
        this.unitsConsumed = unitsConsumed;
    }

    public void setCustomerId(int customerId) {
        this.customerId = customerId;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public void setUnitsConsumed(double unitsConsumed) {
        this.unitsConsumed = unitsConsumed;
    }

    public int getCustomerId() {
        return customerId;
    }

    public String getCustomerName() {
        return customerName;
    }

    public double getUnitsConsumed() {
```

```
        return unitsConsumed;
    }

    public double calculateBill() {
        double bill = 0;
        double units = unitsConsumed;
        if (units <= 50) {
            bill = units * 4;
        } else if (units <= 150) {
            bill = 50 * 4 + (units - 50) * 6;
        } else {
            bill = 50 * 4 + 100 * 6 + (units - 150) * 8;
        }
        if (bill > 2000) {
            bill = bill - (bill * 0.15);
        }
        return bill;
    }
}

class GasBillingSystem {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = Integer.parseInt(sc.nextLine());
        Customer[] customers = new Customer[N];
        for (int i = 0; i < N; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            double units = Double.parseDouble(sc.nextLine());
            customers[i] = new Customer(id, name, units);
        }
        for (Customer c : customers) {
            System.out.println("Customer ID: " + c.getCustomerId());
            System.out.println("Customer Name: " + c.getCustomerName());
            System.out.printf("Final Bill: %.1f\n", c.calculateBill());
        }
        sc.close();
    }
}
```

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

3. Problem Statement

You are working as a developer for CityMobile, which wants to build a basic mobile data usage management system.

Each customer has:

A Customer ID (integer)A Customer Name (string)An Initial Data Balance (in GB, double)

The company allows two types of operations:

Recharge – increases the data balance.Usage – decreases the data balance only if enough data is available.

If the usage amount is greater than the available data balance, the usage should not happen, and the balance should remain the same.

You are required to implement this system using:

A class with attributes for customer details.A constructor to initialize customer details.Setter methods to update details if needed.Getter methods to retrieve details.Objects of the class to represent customers.

Finally, display each customer's details after all operations.

Input Format

The first line of input contains an integer N, representing the number of customers.

For each customer:

- The next line contains the Customer ID (integer).
- The following line contains the Customer Name (string).
- The next line contains the Initial Data Balance (double).
- The next line contains the Recharge Amount in GB (double).
- The next line contains the Usage Amount in GB (double).

Output Format

For each customer, print the details in the following format:

Customer ID: <customer_id>

Customer Name: <customer_name>

Final Data Balance: <final_data_balance> GB (The final balance must be rounded to one decimal place.)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1234
Ravi Kumar
5.0
2.0
3.0

Output: Customer ID: 1234
Customer Name: Ravi Kumar
Final Data Balance: 4.0 GB

Answer

```
import java.util.Scanner;

class Customer {
    private int customerId;
    private String customerName;
    private double dataBalance;

    public Customer(int customerId, String customerName, double dataBalance) {
        this.customerId = customerId;
        this.customerName = customerName;
        this.dataBalance = dataBalance;
    }

    public int getCustomerId() {
        return customerId;
    }

    public String getCustomerName() {
```

```
        return customerName;
    }

    public double getDataBalance() {
        return dataBalance;
    }

    public void recharge(double amount) {
        if (amount >= 0) {
            dataBalance += amount;
        }
    }

    public void useData(double amount) {
        if (amount <= dataBalance) {
            dataBalance -= amount;
        }
    }
}

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

        for (int i = 0; i < n; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            double initial = Double.parseDouble(sc.nextLine());
            double recharge = Double.parseDouble(sc.nextLine());
            double usage = Double.parseDouble(sc.nextLine());

            Customer customer = new Customer(id, name, initial);
            customer.recharge(recharge);
            customer.useData(usage);

            System.out.printf("Customer ID: %d\n", customer.getCustomerId());
            System.out.printf("Customer Name: %s\n",
customer.getCustomerName());
            System.out.printf("Final Data Balance: %.1f GB\n",
customer.getDataBalance());
        }
```

```
    }
}
```

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

4. Problem Statement

Anjali is now working as a developer for the City Marathon Association, which wants to build a system to track and find the fastest runner among marathon participants.

Each runner’s record has:

Runner ID (integer)Runner Name (string)An array of times (in minutes) taken in 5 marathon events (integers)

The system must calculate:

The average time of each runner (sum of all times / 5).Identify the fastest runner (the one with the lowest average time).If two or more runners have the same average time, the one with the lower Runner ID is considered the fastest runner.

Anjali has been asked to implement this system using:

A class with attributes for runner details.A constructor to initialize runner details.Getter and Setter methods to retrieve and update runner details if required.A method to calculate the average time.Objects of the class to represent runners.

Finally, display each runner’s details and announce the Fastest Runner.

Input Format

The first line of input contains an integer N (number of runners).

For each runner:

- The next line contains the Runner ID (integer).
- The following line contains the Runner Name (string).

- The next line contains 5 integers separated by spaces (times in minutes for 5 marathon events).

Output Format

For each runner the output prints the following details:

- Runner ID: <runner_id>
- Runner Name: <runner_name>
- Average Time: <average_time>

Finally, print "Fastest Runner: <runner_name> with <average_time> minutes"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001
Ravi Kumar
240 250 245 255 260

Output: Runner ID: 1001
Runner Name: Ravi Kumar
Average Time: 250
Fastest Runner: Ravi Kumar with 250 minutes

Answer

```
import java.util.Scanner;

class Runner {
    private int runnerId;
    private String runnerName;
    private int[] times;
    private int averageTime;
    public Runner(int runnerId, String runnerName, int[] times) {
        this.runnerId = runnerId;
        this.runnerName = runnerName;
        this.times = times;
```

```
        calculateAverageTime();
    }
    private void calculateAverageTime() {
        int sum = 0;
        for (int time : times) {
            sum += time;
        }
        this.averageTime = sum / times.length;
    }
    public int getRunnerId() {
        return runnerId;
    }

    public String getRunnerName() {
        return runnerName;
    }

    public int getAverageTime() {
        return averageTime;
    }
    public void display() {
        System.out.println("Runner ID: " + runnerId);
        System.out.println("Runner Name: " + runnerName);
        System.out.println("Average Time: " + averageTime);
    }
}

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

        Runner fastestRunner = null;

        for (int i = 0; i < n; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            String[] timeStr = sc.nextLine().split(" ");
            int[] times = new int[5];
            for (int j = 0; j < 5; j++) {
                times[j] = Integer.parseInt(timeStr[j]);
            }
```

```
            Runner runner = new Runner(id, name, times);
            runner.display();
            if (fastestRunner == null ||
                runner.getAverageTime() < fastestRunner.getAverageTime() ||
                (runner.getAverageTime() == fastestRunner.getAverageTime()
                 && runner.getRunnerId() < fastestRunner.getRunnerId())) {
                fastestRunner = runner;
            }
        }
        System.out.println("Fastest Runner: " + fastestRunner.getRunnerName() +
                           " with " + fastestRunner.getAverageTime() + " minutes");
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 6_Q1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Elsa subscribes to a premium service with a base monthly cost, a service tax and an extra feature cost. Assist her in writing an inheritance program that takes input for these values and calculates the total monthly cost.

Refer to the below class diagram:

Input Format

The first line of input consists of a double value, representing the base monthly cost.

The second line consists of a double value, representing the service tax.

The third line consists of a double value, representing the extra feature cost.

Output Format

The output prints "Rs. X" where X is a double value, rounded off to two decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10.0
2.5
5.0
Output: Rs. 17.50

Answer

```
import java.util.Scanner;

class Subscription {
    protected double baseMonthlyCost;

    public Subscription(double baseMonthlyCost) {
        this.baseMonthlyCost = baseMonthlyCost;
    }
}

class PremiumSubscription extends Subscription {
    private double serviceTax;
    private double extraFeatureCost;

    public PremiumSubscription(double baseMonthlyCost, double serviceTax,
double extraFeatureCost) {
        super(baseMonthlyCost);
        this.serviceTax = serviceTax;
        this.extraFeatureCost = extraFeatureCost;
    }

    public double calculateMonthlyCost() {
        return baseMonthlyCost + serviceTax + extraFeatureCost;
    }
}
```

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

        double baseMonthlyCost = scanner.nextDouble();
        double serviceTax = scanner.nextDouble();
        double extraFeatureCost = scanner.nextDouble();

        PremiumSubscription premiumSubscription = new
PremiumSubscription(baseMonthlyCost, serviceTax, extraFeatureCost);

        double totalMonthlyCost = premiumSubscription.calculateMonthlyCost();

        System.out.printf("Rs. %.2f%n", totalMonthlyCost);

        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 6_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Alice is managing an online store and wants to implement a program using inheritance to calculate the selling price of products after applying discounts.

Guide her by following the instructions:

Create a base class called Product with a public double attribute price.Create a subclass called DiscountedProduct, which extends Product and includes a private double attribute discount rate. This subclass has a method called calculateSellingPrice() to determine the final selling price after applying the discount.

Formula: Discounted selling price = price * (1 - discount rate)

Input Format

The first line of input consists of a double value p, the initial price of the product.

The second line consists of a double value d, the discount rate.

Output Format

The output prints "Rs. X", where X is a double value, representing the calculated discounted selling price, rounded off to two decimal places.

If the discount rate is greater than 1, print "Not applicable".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 50.00
0.20

Output: Rs. 40.00

Answer

```
import java.util.Scanner;

class Product {
    public double price;

    public Product(double price) {
        this.price = price;
    }
}

class DiscountedProduct extends Product {
    private double discountRate;

    public DiscountedProduct(double price, double discountRate) {
        super(price);
        this.discountRate = discountRate;
    }

    public double calculateSellingPrice() {
        if (discountRate > 1.0) {
            return -1;
```

```
        }
        return price * (1 - discountRate);
    }
}

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

        double initialPrice = scanner.nextDouble();
        double discountRate = scanner.nextDouble();
        DiscountedProduct discountedProduct = new
DiscountedProduct(initialPrice, discountRate);
        double sellingPrice = discountedProduct.calculateSellingPrice();

        if (sellingPrice >= 0) {
            System.out.printf("Rs. %.2f%n", sellingPrice);
        } else {
            System.out.println("Not applicable");
        }
        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 6_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Preethi is working on a project to automate sales tax calculations for items in a store. She wants to create a program that takes the price of an item and the sales tax rate as input and calculates the final price of the item after applying the sales tax.

Write a program using the class SalesTaxCalculator, which contains an overloaded method named calculateFinalPrice to handle both integer and double inputs. The program should also include a Main class that takes user input, calls the appropriate method from SalesTaxCalculator, and prints the final price of the item.

Formula Used: Final price = price + ((price * sales tax rate) / 100)

Input Format

The first line of input consists of an integer price (the price of the item for integer inputs).

The second line of input consists of an integer taxRate (the sales tax rate for integer inputs).

The third line of input consists of a double price (the price of the item for double inputs).

The fourth line of input consists of a double taxRate (the sales tax rate for double inputs).

Output Format

The first line of output prints an integer, representing the final price of the item after applying the sales tax for integer inputs (a and b).

The second line prints a double value, representing the final price of the item after applying the sales tax for double-value inputs (m and n), rounded to two decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 100
10
100.0
5.0

Output: 110
105.00

Answer

```
import java.util.Scanner;

class SalesTaxCalculator {
    public static int calculateFinalPrice(int price, int taxRate) {
        return price + (price * taxRate) / 100;
    }
    public static double calculateFinalPrice(double price, double taxRate) {
        return price + (price * taxRate) / 100;
```

```
    }
}
class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int intPrice = scanner.nextInt();
        int intTaxRate = scanner.nextInt();
        double doublePrice = scanner.nextDouble();
        double doubleTaxRate = scanner.nextDouble();

        int finalPriceInt = SalesTaxCalculator.calculateFinalPrice(intPrice,
intTaxRate);
        double finalPriceDouble =
SalesTaxCalculator.calculateFinalPrice(doublePrice, doubleTaxRate);

        System.out.println(finalPriceInt);
        System.out.format("%.2f", finalPriceDouble);
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 6_Q4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Mr.Kapoor wants to create a program to calculate the volume of a Cuboid and a Cube using method overriding.

Implements a base class Cuboid with attributes for length, width, and height. Include a method calculateVolume() that computes the volume of the cuboid.

Extends the base class with a subclass Cube representing a cube, where all sides are equal. Override the calculateVolume() method in the Cube class to compute the volume of the cube.

The program should take user input for the dimensions of the cuboid and the side length of the cube and display the calculated volumes with two decimal places.

Input Format

The first line of input consists of 3 space-separated double values, representing the cuboid length, width, and height, respectively.

The second line consists of a double value, representing the side length of the cube.

Output Format

The first line of output prints the volume of the cuboid, rounded off to two decimal places.

The second line prints the volume of the cube, rounded off to two decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 60.0 60.0 60.0
50.0
Output: Volume of Cuboid: 216000.00
Volume of Cube: 125000.00

Answer

```
import java.util.Scanner;

class Cuboid {
    double length;
    double width;
    double height;

    public Cuboid(double length, double width, double height) {
        this.length = length;
        this.width = width;
        this.height = height;
    }

    public double calculateVolume() {
        return length * width * height;
```

```
    }
}

class Cube extends Cuboid {
    double side;

    public Cube(double side) {
        super(side, side, side);
        this.side = side;
    }
    @Override
    public double calculateVolume() {
        return side * side * side;
    }
}


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

        double cuboidLength = scanner.nextDouble();
        double cuboidWidth = scanner.nextDouble();
        double cuboidHeight = scanner.nextDouble();

        // Regular object instantiation for Cuboid
        Cuboid cuboid = new Cuboid(cuboidLength, cuboidWidth, cuboidHeight);
        System.out.printf("Volume of Cuboid: %.2f\n", cuboid.calculateVolume());

        double cubeSide = scanner.nextDouble();

        // Upcasting - Using superclass reference for subclass object (DMD)
        Cuboid cube = new Cube(cubeSide); // Upcasting
        System.out.printf("Volume of Cube: %.2f", cube.calculateVolume()); // Calls
Cube's method dynamically

        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 6_Q5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem statement:

Tim was tasked with developing a grocery shopping app. You have a class hierarchy that includes Item, Produce, and OrganicProduce. Your goal is to calculate the total cost of a shopping list, which may contain a mix of regular produce and organic produce items. Additionally, you need to apply discounts to organic items. Apply a 10% discount on organic produce items

Class Hierarchy:

Item: Base class for all items.

Produce: Subclass of Item for regular produce items.

OrganicProduce: Subclass of Produce for organic produce items.

Input Format

The first line of input consists of an integer, 'n'.

For each 'n' item, the user will provide:

- A string 'type' representing the item type ('Regular' or 'Organic').
- A string 'name' represents the item name.
- A double 'price' represents the item price.

Output Format

The output will display the total cost of the shopping list, including discounts on organic items.

Refer to the sample output for format specifications.

Sample Test Case

Input: 1
Regular Banana 1.99
Output: 1.99

Answer

```
import java.util.Scanner;

class Item {
    String name;
    double price;
    public Item(String name, double price) {
        this.name = name;
        this.price = price;
    }
    public double calculateCost() {
        return price;
    }
}
class Produce extends Item {
    public Produce(String name, double price) {
        super(name, price);
    }
```

```
    @Override
    public double calculateCost() {
        return super.calculateCost();
    }
}
class OrganicProduce extends Produce {
    public OrganicProduce(String name, double price) {
        super(name, price);
    }
    @Override
    public double calculateCost() {
        return price * 0.90;
    }
}

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

        int n = sc.nextInt();
        sc.nextLine(); // Consume newline

        double totalCost = 0.0;

        for (int i = 0; i < n; i++) {
            String type = sc.next();
            String name = sc.next();
            double price = sc.nextDouble();

            if (type.equals("Regular")) {
                Item item = new Produce(name, price);
                totalCost += item.calculateCost();
            } else if (type.equals("Organic")) {
                Item item = new OrganicProduce(name, price);
                totalCost += item.calculateCost();
            }
        }

        System.out.printf("%.2f%n", totalCost);
    }
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 6_PAH

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

John is planning a long road trip and wants to calculate the distance his car can travel based on its speed and fuel capacity. As John knows that different cars have different fuel efficiencies, he wants a program that can help him estimate the travel distance for any given car.

To do this, you are tasked with creating a program that calculates the travel distance of a car based on its speed and fuel capacity. The calculation is simple and follows the formula:

Travel Distance = Speed * Fuel Capacity

You need to model this system using a Vehicle class and a Car class. The Vehicle class will have attributes for the speed and fuel capacity, while the Car class will inherit from the Vehicle class and include a method to

calculate the travel distance.

Input Format

The first line of input consists of a double value representing the speed of the car in km/h.

The second line of input consists of a double value representing the fuel capacity of the car in liters.

Output Format

The first line should print "Speed: X km/h", where X is the speed of the car, rounded to two decimal places.

The second line should print "Fuel Capacity: Y liters", where Y is the fuel capacity of the car, rounded to two decimal places.

The third line should print "Travel Distance: Z km", where Z is the total travel distance the car can cover, rounded to two decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10.0
1.0
Output: Speed: 10.00 km/h
Fuel Capacity: 1.00 liters
Travel Distance: 10.00 km

Answer

```
import java.util.Scanner;

class Vehicle {
   double speed;
   double fuelCapacity;

   public Vehicle(double speed, double fuelCapacity) {
      this.speed = speed;
      this.fuelCapacity = fuelCapacity;
```

```
    }
}

class Car extends Vehicle {
    public Car(double speed, double fuelCapacity) {
        super(speed, fuelCapacity);
    }

    public double calculateTravelDistance() {
        return speed * fuelCapacity;
    }
}

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

        double speed = scanner.nextDouble();
        double fuelCapacity = scanner.nextDouble();

        Car car = new Car(speed, fuelCapacity);

        System.out.println("Speed: " + String.format("%.2f", car.speed) + " km/h");
        System.out.println("Fuel Capacity: " + String.format("%.2f", car.fuelCapacity)
+ " liters");
        System.out.println("Travel Distance: " + String.format("%.2f",
car.calculateTravelDistance()) + " km");

        scanner.close();
    }
}
```

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

2. Problem Statement

In a company, each manager has a unique employee ID and a monthly salary. You are required to design a program that will calculate and display the annual(12 months) salary of a manager based on the input details provided by the user.

Implement the solution using a single inheritance approach.

Employee: The base class with attributes name and employeeID.

Manager: The derived class inheriting from Employee, with an additional attribute salary.

Input Format

The first line of input consists of a string name, representing the manager's name.

The second line of input consists of an integer employeeID, representing the manager's employee ID.

The third line of input consists of a double salary, representing the manager's monthly salary.

Output Format

The first line of output prints: Name: <name>

The second line of output prints: Annual Salary: Rs. <annual_salary> (rounded to two decimal places).

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Davis
234
28750.75

Output: Name: Davis
Annual Salary: Rs. 345009.00

Answer

```
import java.util.Scanner;
import java.text.DecimalFormat;

class Employee {
    String name;
```

```
    int employeeID;

    public Employee(String name, int employeeID) {
        this.name = name;
        this.employeeID = employeeID;
    }
}

class Manager extends Employee {
    double salary;

    public Manager(String name, int employeeID, double salary) {
        super(name, employeeID);
        this.salary = salary;
    }

    public double calculateAnnualSalary() {
        return salary * 12;
    }
}
class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DecimalFormat df = new DecimalFormat("0.00");

        String name = scanner.nextLine();
        int employeeID = scanner.nextInt();
        double salary = scanner.nextDouble();

        Manager manager = new Manager(name, employeeID, salary);

        System.out.println("Name: " + manager.name);
        System.out.println("Annual Salary: Rs. " +
df.format(manager.calculateAnnualSalary()));

        scanner.close();
    }
}
```

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

3. Problem Statement

Ram is designing a program to calculate the Body Mass Index (BMI). Your task is to assist him by following the given specifications.

Create a base class BMIcalculator with a method calculateBMI() to compute BMI using the formula weight / (height * height).

Extend the class with a subclass CustomBMIcalculator that overrides the method calculateBMI() to calculate BMI based on custom criteria, assigning categories such as "Underweight," "Normal Weight," "Overweight," or "Obese."

BMI < 18.5, category = "Underweight"BMI >= 18.5 & < 24.9, category = "Normal Weight"BMI >= 25 & < 29.9, category = "Overweight"else category = "Obese"

Implement user input for weight and height and display both the standard and custom BMI calculations.

Input Format

The first line of input consists of a double value, representing the weight in kgs.

The second line consists of a double value, representing the height in meters.

Output Format

The first line of output prints: "Standard BMI Calculation:"

The second line of output prints: "BMI: " followed by the calculated BMI value (to two decimal places).

The third line of output prints: "Custom BMI Calculation:"

The fourth line of output prints: "Category: " followed by the BMI category.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 69.7
2.6
Output: Standard BMI Calculation:
BMI: 10.31
Custom BMI Calculation:
Category: Underweight

Answer

```
import java.util.Scanner;

class BMIcalculator {
    double weight;
    double height;

    public BMIcalculator(double weight, double height) {
        this.weight = weight;
        this.height = height;
    }

    public double calculateBMI() {
        return weight / (height * height);
    }

    public void displayBMI() {
        System.out.printf("BMI: %.2f\n", calculateBMI());
    }
}

class CustomBMIcalculator extends BMIcalculator {

    public CustomBMIcalculator(double weight, double height) {
        super(weight, height);
    }
    @Override
    public double calculateBMI() {
        return super.calculateBMI();
    }

    public void displayCustomBMI() {
        double bmi = calculateBMI();
        String category;
```

```
        if (bmi < 18.5) {
            category = "Underweight";
        } else if (bmi >= 18.5 && bmi < 24.9) {
            category = "Normal Weight";
        } else if (bmi >= 25 && bmi < 29.9) {
            category = "Overweight";
        } else {
            category = "Obese";
        }

        System.out.println("Category: " + category);
    }
}
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        double weight = scanner.nextDouble();
        double height = scanner.nextDouble();

        BMIcalculator bmiCalculator = new BMIcalculator(weight, height);
        System.out.println("Standard BMI Calculation:");
        bmiCalculator.displayBMI();

        CustomBMIcalculator customBMIcalculator = new
CustomBMIcalculator(weight, height);
        System.out.println("Custom BMI Calculation:");
        customBMIcalculator.displayCustomBMI();

        scanner.close();
    }
}
```

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

4. Problem Statement

Sharon, a software developer, is working on a project to automate velocity calculations for various objects. She wants to create a class named VelocityCalculator with overloaded methods calculateVelocity to calculate

the velocity. One method will accept distance in meters and time in seconds as integers, while another will accept distance and time as doubles.

Help her in completing the project.

Formula: Velocity = distance / time

Input Format

The first line of input consists of an integer, representing the distance in meters (for the integer method).

The second line consists of an integer, representing the time in seconds (for the integer method).

The third line consists of a double value, representing the distance in meters (for the double method).

The fourth line consists of a double value, representing the time in seconds (for the double method).

Output Format

The first line prints the velocity calculated using the integer inputs in the format:

Velocity with integer inputs: <velocity> m/s

The second line prints the velocity calculated using the double inputs in the format:

Velocity with double inputs: <velocity> m/s

Note:

The velocity for the double inputs should be printed with two decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 100
10
100.5
10.2

Output: Velocity with integer inputs: 10 m/s
Velocity with double inputs: 9.85 m/s

Answer

```
import java.util.Scanner;

class VelocityCalculator {
    public static int calculateVelocity(int distance, int time) {
        return distance/time;
    }
    public static double calculateVelocity(double distance, double time){
        return distance/time;
    }
}

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

        int distanceInt = scanner.nextInt();
        int timeInt = scanner.nextInt();

        double distanceDouble = scanner.nextDouble();
        double timeDouble = scanner.nextDouble();

        int velocityInt = VelocityCalculator.calculateVelocity(distanceInt, timeInt);
        double velocityDouble =
VelocityCalculator.calculateVelocity(distanceDouble, timeDouble);

        System.out.println("Velocity with integer inputs: " + velocityInt + " m/s");
        System.out.printf("Velocity with double inputs: %.2f m/s", velocityDouble);

        scanner.close();
    }
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 6_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Adams has a reputation company with a great number of employees. He must calculate the salary weekly according to the hourly rate and working hours. Create a program to define a class Employee with attributes name and hourly rate. Create a subclass HourlyEmployee that calculates the weekly salary based on the number of hours worked.

(The first 40 hours are based on the regular hour rate. If the work hours are greater than 40 then the work wage is 1.5 times the hourly rate)

Note: Use Math(Math.max, Math.min) functions .

Example

Input:

Chris

10

45

Output:

Weekly Salary: Rs.475.00

Explanation:

Calculation:

The first 40 hours are paid normally: 40 × 10 = 400.00The extra 5 hours are paid at 1.5 times the hourly rate: 5 × (10×1.5) = 5 × 15 = 75.00Total salary: 400.00 + 75.00 = 475.00

Input Format

The first line of input consists of a string that represents the name of the employee.

The second line consists of a double value that represents the rate for an hour.

The last line consists of an integer that represents the total hours worked.

Output Format

The output displays the total salary of the employee, where salary is rounded to two decimal places in the format: "Weekly Salary: Rs.<double value>".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Dave
10.0
40
Output: Weekly Salary: Rs.400.00

Answer

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

```
import java.text.DecimalFormat;


class Employee {
    String name;
    double hourlyRate;

    public Employee(String name, double hourlyRate) {
        this.name = name;
        this.hourlyRate = hourlyRate;
    }
}

class HourlyEmployee extends Employee {
    int hoursWorked;

    public HourlyEmployee(String name, double hourlyRate, int hoursWorked) {
        super(name, hourlyRate);
        this.hoursWorked = hoursWorked;
    }

    public double calculateWeeklySalary() {
        double regularHours = Math.min(this.hoursWorked, 40);
        double overtimeHours = Math.max(0, this.hoursWorked - 40);
        double regularPay = regularHours * this.hourlyRate;
        double overtimePay = overtimeHours * this.hourlyRate * 1.5;
        return regularPay + overtimePay;
    }
}


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

        String name = scanner.nextLine();
        double hourlyRate = scanner.nextDouble();
        int hoursWorked = scanner.nextInt();

        HourlyEmployee employee = new HourlyEmployee(name, hourlyRate,
hoursWorked);
```

```
        double weeklySalary = employee.calculateWeeklySalary();
        DecimalFormat df = new DecimalFormat("#.00");
        String formattedSalary = df.format(weeklySalary);
        System.out.println("Weekly Salary: Rs." + formattedSalary);
        scanner.close();
    }
}
```

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

2. Problem Statement

Arun wants to calculate the age gap between the grandfather and the son and determine the father's age after 5 years.

Your task is to assist him in developing a program using three classes: GrandFather, Father, and Son, where the GrandFather stores the grandfather's age, the Father extends GrandFather to include the father's age and calculates his age after 5 years, and Son extends Father to include the son's age and calculate the age difference between the grandfather and the son.

Input Format

The input consists of three integers representing the ages of the grandfather, father, and son, one per line.

Output Format

The first line of output prints "Grandfather and son's age gap:" followed by an integer representing the age gap between the grandfather and the son, ending with "years".

The second line prints "Father's Age:" followed by an integer representing the father's age after 5 years, ending with "years".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 50
30
3
Output: Grandfather and son's age gap: 47 years
Father's Age: 35 years

Answer

```
import java.util.Scanner;

import java.util.Scanner;

class GrandFather {
    protected int grandfatherAge;

    public void setGrandfatherAge(int grandfatherAge) {
        this.grandfatherAge = grandfatherAge;
    }
}

class Father extends GrandFather {
    protected int fatherAge;

    public void setFatherAge(int fatherAge) {
        this.fatherAge = fatherAge;
    }

    public int calculateFatherAgeAfter5Years() {
        return fatherAge + 5;
    }
}

class Son extends Father {
    private int sonAge;

    public void setSonAge(int sonAge) {
        this.sonAge = sonAge;
    }

    public int calculateGrandfatherSonAgeDifference() {
        return grandfatherAge - sonAge;
    }
}
```

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

        int grandfatherAge = scanner.nextInt();
        son.setGrandfatherAge(grandfatherAge);

        int fatherAge = scanner.nextInt();
        son.setFatherAge(fatherAge);

        int sonAge = scanner.nextInt();
        son.setSonAge(sonAge);

        System.out.println("Grandfather and son's age gap: "+
son.calculateGrandfatherSonAgeDifference() + " years");

        int fatherAgeAfter5Years = son.calculateFatherAgeAfter5Years();
        System.out.println("Father's Age: " + fatherAgeAfter5Years + " years");
    }
}
```

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

3. Problem Statement

A painter needs to determine the cost to paint different shapes based on their surface area. The program should be designed to handle the area of a sphere and calculate the total painting cost using the following formulas:

Area of sphere: Area = 4 * pi * r² where pi = 3.14Total painting cost: Cost = cost per square meter * area of sphere

The program will consist of three classes:

Shape class: This class should set the shape type and radius.Area class: This class should extend Shape to calculate the area.Cost class: This class should extend Area to calculate the total painting cost.

Input Format

The input consists of a string representing the shape type, a double value

representing the radius, and another double value representing the cost per square meter on each line.

Output Format

For a valid shape type of "Sphere":

- The first line prints: "Area of Sphere is: <calculated_area>" rounded to two decimal places.
- The second line prints: "Cost to paint the shape is: <total_painting_cost>" rounded to two decimal places.

For any other shape types, print: "Invalid type".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Sphere
3.4
5.8
Output: Area of Sphere is: 145.19
Cost to paint the shape is: 842.12

Answer

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

import java.util.Scanner;

```
class Shape {
    protected String shapeType;
    protected double radius;

    public void setShape(String shapeType, Scanner scanner) {
        this.shapeType = shapeType;
        if ("Sphere".equals(shapeType)) {
            this.radius = scanner.nextDouble();
        }
    }
}
```

```
class Area extends Shape {
    protected double area;

    public void calculateArea() {
        if ("Sphere".equals(shapeType)) {
            double pi = 3.14;
            area = 4 * pi * radius * radius;
        }
    }
}

class Cost extends Area {
    private double costPerSqMeter;
    private double totalCost;

    public void setCost(double costPerSqMeter) {
        this.costPerSqMeter = costPerSqMeter;
    }

    public void calculateCost() {
        if ("Sphere".equals(shapeType)) {
            totalCost = costPerSqMeter * area;
            System.out.printf("Area of Sphere is: %.2f\n", area);
            System.out.printf("Cost to paint the shape is: %.2f\n", totalCost);
        } else {
            System.out.println("Invalid type");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String s = scanner.next();
        Cost shape = new Cost();
        shape.setShape(s, scanner);
        double costToPaint = scanner.nextDouble();
        shape.calculateArea();
        shape.setCost(costToPaint);
        shape.calculateCost();
    }
}
```

Status : Correct *Marks : 10/10*

4. Problem Statement

Mary is managing a business and wants to analyze its profitability. She operates both a regular business model and a seasonal business model. To assess profitability, she uses a program that calculates and compares the profit margins for both models based on revenue and cost.

The program defines:

BusinessUtility class with a method calculateMargin(double revenue, double cost).SeasonalBusinessUtility (inherits from BusinessUtility) and overrides calculateMargin(double revenue, double cost), adding a seasonal adjustment of 10% to the base margin.ProfitabilityChecker class with a method checkProfitability(double regularMargin), which prints "Business is profitable." if the regular margin is 10% or more, otherwise prints "Business is not profitable.".

Mary inputs revenue and cost, and the program compute and display the regular and seasonal margins using:

Margin = ((Revenue − Cost) / Revenue) × 100

Seasonal Margin = Margin + 10

Input Format

The first line of input consists of a double value r, representing the revenue.

The second line consists of a double value c, representing the cost.

Output Format

The first line prints a double value, representing the regular profit margin, rounded to two decimal places, in the format: "Regular Margin: X. XX%", where X.XX denotes the calculated regular margin.

The second line prints a double value, representing the seasonal profit margin, rounded to two decimal places, in the format: "Seasonal Margin: X. XX%", where X.XX denotes the calculated seasonal margin.

The third line prints a string, indicating whether the business is profitable or not profitable, based on the regular margin.

If the regular margin is less than 10, print "Business is not profitable.". If it is 10 or greater, print "Business is profitable."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1000.0
800.0
Output: Regular Margin: 20.00%
Seasonal Margin: 30.00%
Business is profitable.

Answer

```
import java.util.Scanner;

class BusinessUtility {
    public double calculateMargin(double revenue, double cost) {
        return ((revenue - cost) / revenue) * 100;
    }
}

class SeasonalBusinessUtility extends BusinessUtility {
    @Override
    public double calculateMargin(double revenue, double cost) {
        double baseMargin = super.calculateMargin(revenue, cost);
        return baseMargin + 10;
    }
}

class ProfitabilityChecker {
    public void checkProfitability(double regularMargin) {
        if (regularMargin >= 10) {
            System.out.println("Business is profitable.");
        } else {
            System.out.println("Business is not profitable.");
        }
```

```
    }
}

class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double revenue = scanner.nextDouble();
        double cost = scanner.nextDouble();
        BusinessUtility business = new BusinessUtility();
        SeasonalBusinessUtility seasonalBusiness = new
SeasonalBusinessUtility();
        double regularMargin = business.calculateMargin(revenue, cost);
        double seasonalMargin = seasonalBusiness.calculateMargin(revenue,
cost);

        System.out.printf("Regular Margin: %.2f%%\n", regularMargin);
        System.out.printf("Seasonal Margin: %.2f%%\n", seasonalMargin);

        ProfitabilityChecker checker = new ProfitabilityChecker();
        checker.checkProfitability(regularMargin);
        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 7_Q1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement:

Rajiv is analyzing the energy consumption in his household and wants to calculate the total cost based on the daily energy usage. He is given the rate per unit of electricity and the energy consumed for multiple days. To structure this calculation efficiently, he decides to use an interface-based approach.

Implement an interface CostCalculator with the necessary methods to retrieve energy details and compute the cost. The calculations should be handled in the EnergyConsumptionTracker class, while the EnergyConsumptionApp class should only handle input and output.

Formula

Energy Cost for one day = Energy Consumed per day * Rate Per Unit

Input Format

The first line of input consists of the rate per unit as an 'R' (a double value).

The second line of input consists of the number of days 'N' (an integer).

The third line of input consists of the daily energy consumption values for each day 'D" (double values), separated by space.

Output Format

The first line of the output prints: "Day-wise Energy Cost:"

The next N lines of the output print the day-wise energy costs(double type) and the total energy cost (double type) in Indian Rupees in the following format: "Day [day_number]: Rs. [energy_cost]"

The last line of the output prints: "Total Energy Cost: Rs. [total_cost]"

Note: energy_cost and total_cost are rounded off to two decimal points

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 0.01
3
10.0 20.0 30.0
Output: Day-wise Energy Cost:
Day 1: Rs. 0.10
Day 2: Rs. 0.20
Day 3: Rs. 0.30
Total Energy Cost: Rs. 0.60

Answer

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

```
interface CostCalculator
{

    void getEnergyDetails(Scanner scanner);
    void calculateAndDisplayCost();

}

class EnergyConsumptionTracker implements CostCalculator

{

    private double ratePerUnit;
    private int numDays;
    private double[] energyConsumptionArray;

    public EnergyConsumptionTracker(double ratePerUnit, int numDays)

{


        this.ratePerUnit = ratePerUnit;
        this.numDays = numDays;
        this.energyConsumptionArray = new double[numDays];


}

    public void getEnergyDetails(Scanner scanner)

{


        for (int i = 0; i < numDays; i++)

{


            energyConsumptionArray[i] = scanner.nextDouble();
```

```
}

}

   public void calculateAndDisplayCost()

{


      double totalCost = 0;
      System.out.println("Day-wise Energy Cost:");
      for (int i = 0; i < numDays; i++)

{


         double energyCost = energyConsumptionArray[i] * ratePerUnit;
         totalCost += energyCost;
         System.out.printf("Day %d: Rs. %.2f\n", i + 1, energyCost);


}
      System.out.printf("Total Energy Cost: Rs. %.2f\n", totalCost);


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

      double ratePerUnit = scanner.nextDouble();
      int numDays = scanner.nextInt();

      CostCalculator tracker = new EnergyConsumptionTracker(ratePerUnit,
numDays);

      tracker.getEnergyDetails(scanner);
      tracker.calculateAndDisplayCost();
```

```
        scanner.close();
    }
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 7_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Jaheer is working on a health monitoring system to help individuals calculate their Body Mass Index (BMI). He has implemented a basic BMI calculator and an interface called HealthCalculator. It should have a method called calculateBMI.

You are tasked with creating a program that takes weight and height as input, calculates the BMI using the BMICalculator class, and displays the result. If the height or weight is less than or equal to zero, then return -1.

Formula: BMI = weight / (height * height)

Input Format

The first line of input consists of a double value W, the person's weight in kilograms.

The second line consists of a double value H, the height of the person in meters.

Output Format

The output displays "BMI: " followed by a double value, representing the calculated BMI, rounded off to two decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 70.0
1.75
Output: BMI: 22.86

Answer

```
import java.util.Scanner;

// You are using
import java.util.Scanner;

interface HealthCalculator

{

   double calculateBMI(double weight, double height);

}

class BMICalculator implements HealthCalculator

{


   public double calculateBMI(double weight, double height)

{
```

```
        if (weight <= 0 || height <= 0)
{


            return -1;


}
        return weight / (height * height);


}

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

        double weight = scanner.nextDouble();
        double height = scanner.nextDouble();

        BMICalculator bmiCalculator = new BMICalculator();

        double bmi = bmiCalculator.calculateBMI(weight, height);

        System.out.printf("BMI: %.2f\n", bmi);


        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 7_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

A financial analyst, Alex, needs a program to calculate simple interest for various financial transactions. He requires a straightforward tool that takes in the principal amount, interest rate, and time in years and computes the interest.

The formula to be used is: Interest = Principal × Rate × Time / 100

Implement this functionality using the InterestCalculator interface and the SimpleInterestCalculator class.

Input Format

The first line of input consists of the principal amount P as a double value.

The second line of input consists of the annual interest rate r as a double value.

The third line of input consists of the number of years t as a positive integer, which is an integer value.

Output Format

The output displays the calculated simple interest in the following format: "Simple Interest: [interest_value]", Here, [interest_value] should be replaced with the actual interest value calculated by the program.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1000.00
5.00
2
Output: Simple Interest: 100.0

Answer

```
import java.util.Scanner;

// You are using Java
import java.util.Scanner;

interface InterestCalculator
{

    double simpleInterest(double principal, double rate, int time);

}

class SimpleInterestCalculator implements InterestCalculator

{

    public double simpleInterest(double principal, double rate, int time)
```

```
{

        double interest = (principal * rate * time) / 100.0;
        return interest;


}

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

        double principal = scanner.nextDouble();

        double rate = scanner.nextDouble();

        int time = scanner.nextInt();

        InterestCalculator calculator = new SimpleInterestCalculator();

        double interest = calculator.simpleInterest(principal, rate, time);

        System.out.println("Simple Interest: " + interest);
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 7_Q4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Maria, a software developer, is working on an inventory management system project using Java that utilizes an inventory interface to manage a store's products.

The interface should define two methods: addProduct, which adds a product by accepting its name, price, and quantity, and calculateTotalValue, which computes the total value of all products in the inventory. Implement the interface in a class called SimpleInventory, which internally manages a list of Product objects.

Each Product object should encapsulate the product's name, price, and quantity and include a method to calculate its value as price × quantity. The system should allow users to dynamically add products to the inventory and calculate the total value of all products stored.

Help Maria achieve the task.

Input Format

The first line of input consists of an integer to choose one of the following options:

1 - to add a product to the inventory.

2 - to calculate and view the total inventory value.

3 - to exit the program.

For Choice 1 (Add Product):

The next input line is the string representing the product name as a string (single or multi-word, without quotes).

The next line is a double value representing the price as a decimal value

The next line is an integer value representing the quantity as an integer

For Choices 2 and 3, no additional input is required

Output Format

The output displays the results of the commands as follows:

- For the addProduct command, the program should display "Product added to inventory."
- For choice 2, the program should display "Total inventory value [totalvalue]. "The total value should be displayed with one decimal place. If there is no product in the inventory, print the total as 0.0.
- For choice 3, the program should exit

If the choice is not 1, 2, or 3, then print "Invalid choice. Please select a valid option (1/2/3).".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1
Laptop
800.0
3
2
5
3
Output: Product added to inventory.
Total inventory value: $2400.0
Invalid choice. Please select a valid option (1/2/3).

Answer

```
import java.util.Scanner;

class Product

{


   String name;
   double price;
   int quantity;

   Product(String name,double price,int quantity)

{


      this.name = name;
      this.price = price;
      this.quantity = quantity;


}

   double getValue()

{
```

```
        return price * quantity;

    }

    }

    interface Inventory

    {


       void addProduct(String name, double price,int quantity);
       double calculateTotalValue();

    }
    class SimpleInventory implements Inventory

    {


       Product[] products;
       int count;

       SimpleInventory(int capacity)

    {


          products = new Product[capacity];
          count = 0;


    }
       public void addProduct(String name, double price, int quantity)

    {


          if(count < products.length)
    {
```

```
            products[count++] = new Product(name, price, quantity);
            System.out.println("Product added to inventory.");

    }else
    {

            System.out.println("Inventory is full. Cannot add more products.");

    }

    }
       public double calculateTotalValue()
    {

          double tot = 0.0;
          for(int i = 0; i < count; i++)
    {

             tot += products[i].getValue();

    }
          return Math.round(tot * 10.0)/10.0;

    }
    }
    public class Main {
```

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Inventory inventory = new SimpleInventory(10);
        while (true) {
            int choice = scanner.nextInt();
            if (choice == 1) {
                scanner.nextLine();
                String productName = scanner.nextLine();
                double price = scanner.nextDouble();
                int quantity = scanner.nextInt();
                inventory.addProduct(productName, price, quantity);
            } else if (choice == 2) {
                double totalValue = inventory.calculateTotalValue();
                System.out.println("Total inventory value: $" + totalValue);
            } else if (choice == 3) {
                break;
            } else {
                System.out.println("Invalid choice. Please select a valid option
(1/2/3).");
            }
        }
        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 7_Q5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Raj is curious about how old he is in the current year.

He has asked you to create a simple program that calculates a person's age based on their birth year. You decide to implement this functionality using the AgeCalculator interface and the HumanAgeCalculator class.

Note: The current year is 2024. Calculate the current age by using the formula: current year - birth year.

Input Format

The input consists of an integer representing the birth year.

Output Format

The output displays "You are X years old." where X is an integer representing the calculated age based on the entered birth year.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1934
Output: You are 90 years old.

Answer

```
import java.util.Scanner;
// You are using Java
interface AgeCalculator
{

    int calculateAge(int birthYear);

}
class HumanAgeCalculator implements AgeCalculator
{

    public int calculateAge(int birthYear)
{

        return (2024 - birthYear);

}

}
class AgeCalculatorApp {
```

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

        AgeCalculator ageCalculator = new HumanAgeCalculator();

        int birthYear = scanner.nextInt();
        int age = ageCalculator.calculateAge(birthYear);

        System.out.println("You are " + age + " years old.");
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 7_PAH

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement:

Alice has been tasked with implementing a simple calculator interface and a corresponding class for performing basic addition and subtraction operations. The task is to create an interface called Calculator with two methods: add and subtract. The add method should take two numbers as input and return their sum, while the subtract method should take two numbers as input and return their difference.

Implement a class called SimpleCalculator that implements the Calculator interface. This class should provide the functionality for adding and subtracting numbers. Write a code that satisfies the above requirements.

Input Format

The first line of input consists of a single integer, representing the choice

If the choice is 1 or 2, the next two lines consist of 2 double values, representing the numbers to do addition or subtraction.

Output Format

The output prints a float-value with one decimal value representing the sum of two number or difference of two number.

Refer to the sample output for format specification.

Sample Test Case

Input: 1
5.5
3.5
Output: Result: 9.0

Answer

```
import java.util.Scanner;

interface Calculator
{


  double add(double n1, double n2);
  double subtract(double n1, double n2);

}

class SimpleCalculator implements Calculator

{


   public double add(double n1, double n2)

{
```

```
        return n1+n2;


    }
       public double subtract(double n1, double n2)
    {


        return n1 - n2;

    }

    }

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

          SimpleCalculator calculator = new SimpleCalculator();

          int choice = scanner.nextInt();

          if (choice == 1) {
             double num1 = scanner.nextDouble();
             double num2 = scanner.nextDouble();
             double result = calculator.add(num1, num2);
             System.out.println("Result: " + result);
          } else if (choice == 2) {
             double num1 = scanner.nextDouble();
             double num2 = scanner.nextDouble();
             double result = calculator.subtract(num1, num2);
             System.out.println("Result: " + result);
          } else {
             System.out.println("Invalid choice. Please choose 1 for addition or 2 for
    subtraction.");
          }
```

```
        scanner.close();
    }
}
```

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

2. Problem Statement

Develop a program for managing employee information that caters to both full-time and part-time employees. The program should be capable of computing the salary for each category of employee and presenting their particulars. To achieve this, create two classes, FullTimeEmployee and PartTimeEmployee, that adhere to the Employee interface.

The program is expected to accept input data, including the name and monthly salary for full-time employees, as well as the name, hourly rate, and hours worked for part-time employees. Subsequently, it should calculate and exhibit the employee details and their respective salaries.

For Full-Time employees, the annual salary should be calculated as 12 times the monthly salary.

For Part-Time employees, the salary calculation should be based on the formula: hourly rate * hours worked.

Input Format

The first line of input should be a string representing the name of a full-time employee.

The second line of input should be an integer representing the monthly salary of the full-time employee.

The third line of input should be a string representing the name of a part-time employee.

The fourth line of input should be an integer representing the hourly rate of the part-time employee.

The fifth line of input should be an integer representing the number of hours

worked by the part-time employee.

Output Format

The output displays the following details:

Full-Time Employee Details:

Name: [Full-Time Employee Name] (string)

Monthly Salary: $[Monthly Salary] (integer)

Annual Salary: $[12 times Monthly Salary] (integer)

Part-Time Employee Details:

Name: [Part-Time Employee Name] (string)

Hourly Rate: $[Hourly Rate] (integer)

Hours Worked: [Hours Worked] hours (integer)

Monthly Salary: $[Calculated Monthly Salary] (integer)

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: John Smith
15000
Mary Johnson
100
100
Output: Full-Time Employee Details:
Name: John Smith

Monthly Salary: $15000
Annual Salary: $180000

Part-Time Employee Details:
Name: Mary Johnson
Hourly Rate: $100
Hours Worked: 100 hours
Monthly Salary: $10000

Answer

```
import java.util.Scanner;


interface Employee
{


   void displayDetails();
   //void displayDetails(String partTimeName, int hourlyRate, int hoursWorked);

}

class FullTimeEmployee implements Employee

{


   String fullName;
   int fullTimeSalary;
   FullTimeEmployee(String fullName, int fullTimeSalary)

{


      this.fullName = fullName;
      this.fullTimeSalary = fullTimeSalary;


}

   int calculateSalary()
{
```

```
        return fullTimeSalary * 12;

    }
    public void displayDetails()
    {

        System.out.println("Full-Time Employee Details:");
        System.out.println("Name: " + fullName);
        System.out.println("Monthly Salary: $" + fullTimeSalary);
        System.out.println("Annual Salary: $" + calculateSalary());

    }

    }

    class PartTimeEmployee implements Employee

    {

        String partTimeName;
        int hourlyRate;
        int hoursWorked;

        PartTimeEmployee(String partTimeName, int hourlyRate, int hoursWorked)

    {

        this.partTimeName = partTimeName;
        this.hourlyRate = hourlyRate;
        this.hoursWorked = hoursWorked;

    }
```

```
    int calculateSalary()
{

        return hourlyRate * hoursWorked;


}
    public void displayDetails()

{


        System.out.println("Part-Time Employee Details:");
        System.out.println("Name: " + partTimeName);
        System.out.println("Hourly Rate: $" + hourlyRate);
        System.out.println("Hours Worked: " + hoursWorked + " hours");
        System.out.println("Monthly Salary: $" + calculateSalary());


}

}
class EmployeeInheritanceDemo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String fullName = scanner.nextLine();
        int fullTimeSalary = scanner.nextInt();
        scanner.nextLine();
        String partTimeName = scanner.nextLine();
        int hourlyRate = scanner.nextInt();
        int hoursWorked = scanner.nextInt();
        FullTimeEmployee fullTimeEmployee = new FullTimeEmployee(fullName,
fullTimeSalary);
        PartTimeEmployee partTimeEmployee = new
PartTimeEmployee(partTimeName, hourlyRate, hoursWorked);
        fullTimeEmployee.displayDetails();
        System.out.println();
        partTimeEmployee.displayDetails();
        scanner.close();
    }
```

}

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

3. Problem Statement

Sophia is developing a matrix analysis tool for a data analytics company. The tool needs to analyze square matrices and extract insights from the matrix diagonals.

To organize the code properly, Sophia creates an interface named Matrix that declares a method for finding the smallest and largest elements along the principal and secondary diagonals of the matrix.

Sophia then creates a class named MatrixAnalyzer that implements the Matrix interface. This class provides the logic to process a given square matrix and print:

The smallest and largest elements in the principal diagonal (from top-left to bottom-right).The smallest and largest elements in the secondary diagonal (from top-right to bottom-left).

Your task is to implement the Matrix interface and the MatrixAnalyzer class. The main driver program (in the class Main) will read the input matrix, create an instance of MatrixAnalyzer, and invoke its method to display the results.

Input Format

The first line contains an integer n, representing the size of the square matrix.

The next n lines each contain n integers separated by spaces, representing the elements of the matrix.

Output Format

The output prints the four lines:

"Smallest Element - 1: <smallest element in the principal diagonal>" (integer)

"Largest Element - 1: <largest element in the principal diagonal>" (integer)

"Smallest Element - 2: <smallest element in the secondary diagonal>" (integer)

"Largest Element - 2: <largest element in the secondary diagonal>" (integer)

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
7 8 9 0 1
2 3 4 5 6
5 4 2 0 8
23 5 6 8 9
12 5 6 7 32
Output: Smallest Element - 1: 2
Largest Element - 1: 32
Smallest Element - 2: 1
Largest Element - 2: 12

Answer

```
import java.util.Scanner;

// You are using Java
interface Matrix
{

   void diagonalsMinMax(int[][] matrix);

}

class MatrixAnalyzer implements Matrix

{

   public void diagonalsMinMax(int[][] matrix)
```

```
{


        int n = matrix.length;

        int minPrin = matrix[0][0];
        int maxPrin = matrix[0][0];
        int minSec = matrix[0][n - 1];
        int maxSec = matrix[0][n - 1];

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

{


            int prinEle = matrix[i][i];
            if(prinEle < minPrin) minPrin = prinEle;
            if(prinEle > maxPrin) maxPrin = prinEle;

            int secEle = matrix[i][n - 1 - i];
            if(secEle < minSec) minSec = secEle;
            if(secEle > maxSec) maxSec = secEle;


}
        System.out.println("Smallest Element - 1: " + minPrin);
        System.out.println("Largest Element - 1: " + maxPrin);
        System.out.println("Smallest Element - 2: " + minSec);
        System.out.println("Largest Element - 2: " + maxSec);


}

}
public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      int[][] matrix = new int[n][n];
      for (int i = 0; i < n; i++) {
         for (int j = 0; j < n; j++) {
            matrix[i][j] = sc.nextInt();
```

```
        }
    }
        MatrixAnalyzer analyzer = new MatrixAnalyzer();
        analyzer.diagonalsMinMax(matrix);
    }
}
```

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

4. Problem Statement

Oviya is fascinated by automorphic numbers and wants to create a program to determine whether a given number is an automorphic number or not.

An automorphic number is a number whose square ends with the same digits as the number itself. For example, 25 = (25)2 = 625

Oviya has defined two interfaces: NumberInput for taking user input and AutomorphicChecker for checking if a given number is automorphic. The class AutomorphicNumber implements both interfaces.

Help her complete the task.

Input Format

The input consists of a single integer n.

Output Format

If the input number is an automorphic number, print "n is an automorphic number". Otherwise, print "n is not an automorphic number".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 25
Output: 25 is an automorphic number

Answer

```
import java.util.Scanner;

interface NumberInput
{


   int getInput();

}
interface AutomorphicChecker

{


   boolean checkAutomorphic(int inputNumber);

}

class AutomorphicNumber implements NumberInput,AutomorphicChecker

{


   public int getInput()

{


      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      return n;


}
   public boolean checkAutomorphic(int inputNumber)

{


      int sq = inputNumber * inputNumber;
```

```
        String num = Integer.toString(inputNumber);
        String sqNum = Integer.toString(sq);

        return sqNum.endsWith(num);


    }

    }
    public class Main {
       public static void main(String[] args) {
          AutomorphicNumber automorphicNumber = new AutomorphicNumber();
          int inputNumber = automorphicNumber.getInput();

          boolean isAutomorphic =
    automorphicNumber.checkAutomorphic(inputNumber);

          if (isAutomorphic) {
             System.out.println(inputNumber+" is an automorphic number");
          } else {
             System.out.println(inputNumber+" is not an automorphic number");
          }
       }
    }
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 7_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

John is developing a car loan calculator and has structured his program using two interfaces, Principal and InterestRate, defining methods for principal and interest rate retrieval.

The Loan class implements these interfaces, taking principal and annual interest rates as parameters. User input is solicited for these values, and the program ensures their validity before performing calculations. If input values are invalid (less than or equal to zero), an error message is displayed.

Note: Total interest = principal * interest rate * years

Input Format

The first line of input consists of a double value P, representing the principal.

The second line consists of a double value R, representing the annual interest rate.

The third line consists of an integer value N, representing the loan duration in years.

Output Format

If the input values are valid, print "Total interest paid: Rs. " followed by a double value, representing the total interest paid, rounded off to two decimal places.

If the input values are invalid (negative or zero values for principal, annual interest rate, or loan duration), print "Invalid input values!".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 20000.00
0.05
5
Output: Total interest paid: Rs.5000.00

Answer

```
import java.util.Scanner;

interface Principal
{


   double getPrincipal();

}

interface InterestRate

{
```

```
    double getInterestRate();
}

class Loan implements Principal, InterestRate

{


    private double principal;
    private double interestRate;

    Loan(double principal, double interestRate)
{


        this.principal = principal;
        this.interestRate = interestRate;


}

    public double getPrincipal()

{


        return principal;


}

    public double getInterestRate()

{


        return interestRate;
```

```
}
    public double calculateTotalInterest(int years)
{

        return principal * interestRate * years;


}

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

        double carPrice = scanner.nextDouble();

        double annualInterestRate = scanner.nextDouble();

        int loanDuration = scanner.nextInt();

        if (carPrice <= 0 || annualInterestRate <= 0 || loanDuration <= 0) {
            System.out.println("Invalid input values!");
            return;
        }

        Loan carLoan = new Loan(carPrice, annualInterestRate);
        double totalInterest = carLoan.calculateTotalInterest(loanDuration);

        System.out.printf("Total interest paid: Rs.%.2f%n", totalInterest);
    }
}
```

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

2. Problem Statement:

Ray is developing a tax calculation program in Java. The program includes

an interface named TaxCalculator with a method to calculate tax based on salary. The SimpleTaxCalculator class implements this interface and determines the tax to be paid based on the salary amount using progressive tax slabs.

Your task is to implement this system. The program first takes an integer T representing the number of test cases, followed by T salary values. For each salary, calculate the total tax to be paid based on the following progressive tax rules:

For the first 50,000 of salary, the tax rate is 5%.For the next 50,000 (i.e., from 50,001 to 1,00,000), the tax rate is 10%.For any amount above 1,00,000, the tax rate is 20%. (That is, only the amount above 1,00,000 is taxed at 20%.)

Example

Input

3

78000

110000

23000

Output

5300

9500

1150

Explanation

For Salary Rs. 78,000

Tax = 0.1 * (78,000 - 50,000) + 0.05 * 50,000 = 5,300

For Salary Rs. 1,10,000

Tax = 0.2 * (110000 - 100000)+ 0.1 * 50,000 + 0.05 * 50,000 = 9,500

For Salary Rs. 23,000

Tax = 0.05 * 23,000 = 1,150

Input Format

The first line of the input consists of an integer, T, representing the number of test cases.

The next T lines of the input consist of a single integer, representing the annual salary of an individual, separated by a line.

Output Format

The output displays the calculated tax as an integer for each test case, separated by a line.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 2
100
300
Output: 5
15

Answer

```
import java.util.Scanner;

// You are using Java
interface TaxCalculator

{


   int calculateTax(int salary);

}

class SimpleTaxCalculator implements TaxCalculator
```

```
{
    @Override
    public int calculateTax(int salary)
{

        long tax = 0;

        if (salary > 100000)

{

            long taxableAmount = salary - 100000;
            tax += taxableAmount * 0.20;
            salary = 100000;


}

        if (salary > 50000)

{

            long taxableAmount = salary - 50000;
            tax += taxableAmount * 0.10;
            salary = 50000;


}

        if (salary > 0)

{

            tax += salary * 0.05;
```

```
}
        return (int) Math.round(tax);


}

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

        TaxCalculator taxCalculator = new SimpleTaxCalculator();

        for (int i = 0; i < T; i++) {
            int salary = scanner.nextInt();
            int tax = taxCalculator.calculateTax(salary);
            System.out.println(tax);
        }

        scanner.close();
    }
}
```

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

3. Problem Statement:

Sam is developing a geometry application and needs a class for trapezoid calculations. Create a "Trapezoid" class implementing a "ShapeInput" interface with a method to input trapezoid dimensions.

Also, implement a "ShapeCalculator" interface with methods to compute area and perimeter. In the "Main" class, instantiate Trapezoid, gather user input, and display the calculated area and perimeter with two decimal places.

Note

Area of Trapezoid = (1/2) * (base1 + base2) * height

Perimeter of Trapezoid = base1 + base2 + side1 + side2

Input Format

The first line of input is a double-point value representing base1 of the trapezoid.

The second line of input is a double-point value representing base2 of the trapezoid.

The third line of input is a double-point value representing the height of the trapezoid.

The fourth line of input is a double-point value representing side1 of the trapezoid.

The fifth line of input is a double-point value representing side2 of the trapezoid.

Output Format

The output displays the two lines of the calculated area (double type) and perimeter (double type) of the trapezoid, each rounded to two decimal places in the following format:

"Area of the Trapezoid: <<calculated area>>".

Perimeter of the Trapezoid: <<calculated perimeter>>".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1.0
2.0
1.0
3.0
1.0

Output: Area of the Trapezoid: 1.50
Perimeter of the Trapezoid: 7.00

Answer

```
import java.util.Scanner;


interface ShapeInput
{


   void getInput();

}
interface ShapeCalculator
{


   double calculateArea();
   double calculatePerimeter();

}

class Trapezoid implements ShapeInput,ShapeCalculator

{


   private double b1,b2,hg,s1,s2;

   public void getInput()

{


      Scanner sc = new Scanner(System.in);
      b1 = sc.nextDouble();
      b2 = sc.nextDouble();
      hg = sc.nextDouble();
      s1 = sc.nextDouble();
      s2 = sc.nextDouble();
```

```
}
   public double calculateArea()
{

      return (0.5) * (b1 + b2) * hg;

}
   public double calculatePerimeter()
{

      return b1 + b2 + s1 + s2;

}

}
public class Main {
   public static void main(String[] args) {
      Trapezoid trapezoid = new Trapezoid();
      trapezoid.getInput();

      double area = trapezoid.calculateArea();
      double perimeter = trapezoid.calculatePerimeter();

      System.out.println("Area of the Trapezoid: " + String.format("%.2f", area));
      System.out.println("Perimeter of the Trapezoid: " + String.format("%.2f",
perimeter));
   }
}
```

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

4. Problem Statement:

Rathish is planning a road trip and needs a program to convert speeds between miles per hour (MPH) and kilometers per hour (KPH).

Create an interface, SpeedConverter, with a method convertSpeed(double mph). Implement the interface with MPHtoKPHConverter class, allowing Rathish to input MPH and receive the converted speed in KPH, rounded to two decimal points.

Formula: Speed in KPH = 1.60934 * Speed in MPH.

Input Format

The input consists of a single double-point number representing the speed in miles per hour (MPH).

Output Format

The output displays the converted speed (double-point number) in kilometers per hour (KPH) rounded off to two decimal points in the following format:

"Speed in KPH: <<converted speed>>".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1.0
Output: Speed in KPH: 1.61

Answer

```
import java.util.Scanner;


interface SpeedConverter

{

    double convertSpeed(double mph);
```

```
}
class MPHtoKPHConverter implements SpeedConverter
{

    public double convertSpeed(double mph)
{

        return 1.60934 * mph;
}
}
public class Main
{

    public static void main(String[] args)
{

        Scanner scanner = new Scanner(System.in);
        double mph = scanner.nextDouble();
        // 1.0 ≤ Speed in MPH ≤ 100.0 as per constraints
        MPHtoKPHConverter converter = new MPHtoKPHConverter();
        double kph = converter.convertSpeed(mph);
        System.out.printf("Speed in KPH: %.2f%n", kph);
        scanner.close();

}
}
```

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

        double speedInMPH = scanner.nextDouble();

        SpeedConverter converter = new MPHtoKPHConverter();

        double speedInKPH = converter.convertSpeed(speedInMPH);

        System.out.printf("Speed in KPH: %.2f\n", speedInKPH);

        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 8_Q1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Write a program to validate the email address and display suitable exceptions if there is any mistake.

Create 3 custom exception classes as below

DotExceptionAtTheRateExceptionDomainException

A typical email address should have a " . " character, and a "@" character, and also the domain name should be valid. Valid domain names for practice be 'in', 'com', 'net', or 'biz'.

Display Invalid Dot usage, Invalid @ usage, or Invalid Domain message based on email id.

Get the email address from the user, validate the email by checking the

above-mentioned criteria, and print the validity status of the input email address.

Input Format

The first line of input contains the email to be validated.

Output Format

The output prints a Valid email address or an Invalid email address along with the suitable exception

If email ends with . or contains not exactly one . after @, it throws:

DotException: Invalid Dot usage

Invalid email address

If @ appears not exactly once, it throws:

AtTheRateException: Invalid @ usage

Invalid email address

If the part after the last dot is not among accepted domains:

DomainException: Invalid Domain

Invalid email address

If all conditions satisfied then print:

Valid email address

Refer to the sample input and output for format specifications.

Sample Test Case

Input: sample@gmail.com
Output: Valid email address

Answer

```
import java.util.*;
class DotException extends Exception
{

   public DotException(String message)
{

      super(message);

}
}
class AtTheRateException extends Exception
{

   public AtTheRateException(String message)
{

      super(message);
```

```
}
}
class DomainException extends Exception
{

   public DomainException(String message)
{

      super(message);

}
}
class EmailValidator
{

   public static void main(String[] args)
{

      Scanner sc = new Scanner(System.in);
      String email = sc.nextLine();
      sc.close();

      try
{

         validateEmail(email);
         System.out.println("Valid email address");
```

```
}
        catch(DotException e)

{


            System.out.println("DotException: " + e.getMessage());
            System.out.println("Invalid email address");


}
        catch(AtTheRateException e)
{


            System.out.println("AtTheRateException: " + e.getMessage());
            System.out.println("Invalid email address");


}
        catch(DomainException e)

{


            System.out.println("DomainException: " + e.getMessage());
            System.out.println("Invalid email address");


}


}

    public static void validateEmail(String email) throws DotException,
AtTheRateException, DomainException

{
```

```
        if(email.chars().filter(ch -> ch == '@').count() != 1)
{


            throw new AtTheRateException("Invalid @ usage");


}
        if(email.startsWith(".") || email.startsWith("@") || email.endsWith(".") ||
email.endsWith("@"))

{


            throw new DotException("Invalid Dot usage");


}
        if(email.contains("..") || email.contains("@@"))

{


            throw new DotException("Invalid Dot Usage");


}

        String[] parts = email.split("@");
        if(parts.length != 2)

{


            throw new AtTheRateException("Invalid @ usage");


}

        String domainPart = parts[1];
```

```
        if(!domainPart.contains("."))
{

            throw new DotException("Invalid Dot usage");


}

        String extension = domainPart.substring(domainPart.lastIndexOf(".") + 1);

        if(!(extension.equals("com") || extension.equals("in") ||
extension.equals("net") || extension.equals("biz")))
{

            throw new DomainException("Invalid Domain");


}


}

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 8_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Elsa, a busy professional, is using a scheduling application to plan her meetings efficiently. The application requires users to input meeting durations in minutes, ensuring that the duration is a positive integer and does not exceed 240 minutes (4 hours). Elsa needs a program to assist her in scheduling meetings securely with proper exception handling.

Create a Java class named ElsaMeetingScheduler.Implement a custom exception:InvalidDurationException for invalid meeting duration entries.Implement the main method to interactively take user input for a meeting duration.Implement the validateMeetingDuration method to validate the meeting duration based on the specified rules and throw a custom exception if the validation fails.Print appropriate success or error messages based on the meeting duration.

Implement a custom exception, InvalidDurationException, to handle cases where the entered meeting duration does not meet the specified criteria.

Input Format

The input consists of an integer value 'n', representing the meeting duration.

Output Format

The output is displayed in the following format:

If the entered meeting duration meets the specified criteria, the program outputs

"Meeting scheduled successfully!"

If the entered meeting duration is invalid, the program outputs an error message indicating the issue.

"Error: Invalid meeting duration. Please enter a positive integer not exceeding 240 minutes (4 hours)."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 120
Output: Meeting scheduled successfully!

Answer

```
import java.util.*;

class InvalidDurationException extends Exception

{


   public InvalidDurationException(String message)

{
```

```
        super(message);

    }

    }

    class ElsaMeetingScheduler

    {


       public static void validateMeetingDuration(int duration) throws
    InvalidDurationException

    {


           if(duration <= 0 || duration > 240)

    {


               throw new InvalidDurationException("Error: Invalid meeting duration.
    Please enter a positive integer not exceeding 240 minutes (4 hours).");

    }


    }

       public static void main(String[] args)

    {


           Scanner sc = new Scanner(System.in);
           int duration = sc.nextInt();
           sc.close();
```

```
        try
{

            validateMeetingDuration(duration);
            System.out.println("Meeting scheduled successfully!");


}
        catch(InvalidDurationException e)

{

            System.out.println(e.getMessage());


}


}

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 8_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a user registration system, there is a requirement to implement a username validation module. Users attempting to register must adhere to specific criteria for their usernames to be considered valid.

Your task is to develop a program that takes user input for a desired username and validates it according to the following rules:

The username must not contain any spaces.The username must be at least 5 characters long.

Implement a custom exception, InvalidUsernameException, to handle cases where the entered username does not meet the specified criteria.

Input Format

The input consists of a string S, representing the desired username.

Output Format

If the username is valid, print "Username is valid: [S]".

If the username is invalid:

1. If the username is short, print "Invalid Username: Username must be at least 5 characters long"
2. If the username contains spaces, print "Invalid Username: Username cannot contain spaces"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: John
Output: Invalid Username: Username must be at least 5 characters long

Answer

```
import java.util.*;
class InvalidUsernameException extends Exception

{

   public InvalidUsernameException(String message)

{


      super(message);


}

}

class UsernameValidator
```

```
{

    public static void validateUsername(String username) throws
InvalidUsernameException

{


        if(username.contains(" "))

{


            throw new InvalidUsernameException("Invalid Username: Username
cannot contain spaces");


}
        if(username.length() < 5)

{


            throw new InvalidUsernameException("Invalid Username: Username must
be at least 5 characters long");


}


}

    public static void main(String[] args)

{


        Scanner sc = new Scanner(System.in);
        String username = sc.nextLine();
        sc.close();
```

```
    try
{

        validateUsername(username);
        System.out.println("Username is valid: " + username);


}catch(InvalidUsernameException e)

{

        System.out.println(e.getMessage());


}


}

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 8_Q4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

A local municipality is implementing an online voting system for a community event and wants to ensure that only eligible voters (those aged 18 or older) can participate.

Your task is to develop a program that validates the age of individuals attempting to vote online. If the user's age is below 18, the program should throw a custom exception, InvalidAgeException, preventing them from casting their vote. If the input is invalid, catch the appropriate InputMismatchException and print the in-built exception message.

Input Format

The input consists of an integer representing the age.

Output Format

If the age is 18 or older, print "Eligible to vote"

If the age is below 18, print "Exception occurred: InvalidAgeException: Age is not valid to vote"

If there is any other type of exception, print "An error occurred: " followed by the in-built exception message.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 20
Output: Eligible to vote

Answer

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

class InvalidAgeException extends Exception

{

   public InvalidAgeException(String message)

{


      super(message);


}

}

class VotingEligibility
```

```
{
    public static void validateAge(int age) throws InvalidAgeException
{

        if(age < 18)
{

            throw new InvalidAgeException("Age is not valid to vote");
}

}
    public static void main(String[] args)
{

        Scanner sc = new Scanner(System.in);
        try
{

            int age = sc.nextInt();
            validateAge(age);
            System.out.println("Eligible to vote");

}
        catch(InvalidAgeException e)
{
```

```
            System.out.println("Exception occurred: InvalidAgeException: " +
e.getMessage());

        }
            catch(InputMismatchException e)
        {

            System.out.println("An error occurred: " + e);

        }
            catch(Exception e)
        {

            System.out.println("An error occurred: " + e);

        }finally
        {

            sc.close();

        }

        }
        }
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 8_Q5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a file management system, users are required to provide a valid file name when creating new files. The system enforces specific rules for file names to maintain consistency and avoid potential issues. Your task is to implement a Java program named FileNameValidator that takes user input for a file name and validates it according to the specified rules.

Rules for Valid File Name:

The file name must consist of alphanumeric characters (letters and digits) only.The file name must have a minimum length of 3 characters.

Implement a custom exception, FileNameValidator, to handle cases where the entered filename does not meet the specified criteria.

Input Format

The input consists of a string S, representing the desired filename.

Output Format

The output is displayed in the following format:

If the entered file name meets the specified criteria, the program outputs

"Valid file name"

If the entered file name does not meet the criteria and triggers the InvalidFileNameException, the program outputs

"Error: Invalid file name. It must be alphanumeric and have a minimum length of 3 characters."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: myfile123
Output: Valid file name

Answer

```
import java.util.*;
class InvalidFileNameException extends Exception
{

   public InvalidFileNameException(String message)
{

      super(message);

}
```

```
}
class FileNameValidator
{

   public static void validateFileName(String fileName) throws
InvalidFileNameException

{

      if(fileName.length() < 3 || !fileName.matches("[A-Za-z0-9]+"))
{

         throw new InvalidFileNameException("Error: Invalid file name. It must be
alphanumeric and have a minimum length of 3 characters.");


}


}
   public static void main(String[] args)
{

      Scanner sc = new Scanner(System.in);
      String fileName = sc.nextLine();
      sc.close();

      try

{

         validateFileName(fileName);
```

```
            System.out.println("Valid file name");

        }
            catch(InvalidFileNameException e)

        {

                System.out.println(e.getMessage());


        }

    }

    }
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 8_PAH

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

You are tasked to create a program that defines a custom exception GradeException. The program should include a Student class with fields for the student's name, age, and grade. Implement a method in the Student class that checks the grade, and if the grade is below 40, it should throw a GradeException. Otherwise, it should display the student's details.

Input Format

The input consists of three parameters in separate lines:

1. A string representing the student's name.
2. An integer representing the student's age.
3. An integer representing the student's grade.

Output Format

The output will display the student's details if the grade is valid.

If the grade is below 40, the program will display an error message "Grade is below 40".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Alice
20
85
Output: Name: Alice
Age: 20
Grade: 85

Answer

```
import java.util.Scanner;
class GradeException extends Exception

{


   public GradeException(String message)

{


      super(message);


}

}
class Student

{


   String name;
```

```
    int age;
    int grade;
    public Student(String name, int age, int grade)

{


        this.name = name;
        this.age = age;
        this.grade = grade;


}
    public void checkGrade() throws GradeException
{


        if (grade < 40)

{


            throw new GradeException("Grade is below 40");


} else
{


            System.out.println("Name: " + name);
            System.out.println("Age: " + age);
            System.out.println("Grade: " + grade);


}


}
}
```

```
class StudentGradeChecker
{

   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      String name = sc.nextLine();
      int age = sc.nextInt();
      int grade = sc.nextInt();

      Student student = new Student(name, age, grade);
      try

{


         student.checkGrade();


} catch (GradeException e)

{


         System.out.println(e.getMessage());


}


}

}
```

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

2. Problem Statement

Enigma is developing a simple web application that takes a user-input URL, validates it, and throws a custom exception InvalidURLFormatException if the URL does not start with "http://" or "https://".

The main method prompts the user for input, validates the URL, and prints whether it is valid or not.

Input Format

The input consists of a string, representing the URL entered by the user.

Output Format

The output displays one of the following results:

If the entered URL is valid according to the specified format, the program prints:

"[URL] is a valid URL"

If the entered URL is not valid according to the specified format, the program prints:

"Invalid URL format: [URL]"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: http://www.example.com
Output: http://www.example.com is a valid URL

Answer

```
import java.util.Scanner;
class InvalidURLFormatException extends Exception
```

```
{
    public InvalidURLFormatException(String message)
{
        super(message);
}
}
class URLValidator
{
    public static void validateURL(String url) throws InvalidURLFormatException
{
        if (!(url.startsWith("http://") || url.startsWith("https://")))
{
            throw new InvalidURLFormatException("Invalid URL format: " + url);
}
}
public static void main(String[] args)
{
        Scanner sc = new Scanner(System.in);
```

```
        String url = sc.nextLine();

        try

{


            validateURL(url);
            System.out.println(url + " is a valid URL");


} catch (InvalidURLFormatException e)

{


            System.out.println(e.getMessage());


}


}

}
```

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

3. Problem Statement

An HR software system is being developed to process employee payrolls. During payroll processing, the system must ensure that no employee has a negative salary and that no employee's salary exceeds 2,00,000. If either condition occurs, the system should throw a custom exception.

Create a custom exception InvalidSalaryException and a class Employee that processes salary according to the following rules:

If salary < 0, throw InvalidSalaryException with the message: "Salary cannot be negative". If salary > 200000, throw InvalidSalaryException with

the message: "Salary exceeds threshold limit". Otherwise, display: "Salary processed successfully for <empName>: <salary>".

The payroll processing should always display: "Payroll process completed" at the end, regardless of whether an exception occurs.

Input Format

The first line of input contains an integer representing the employee ID.

The second line contains a string representing the employee's name.

The third line contains a floating-point number representing the salary of the employee.

Output Format

If the salary is valid: "Salary processed successfully for <empName>: <salary>"

"Payroll process completed"

If the salary is invalid: "<Exception Message>"

"Payroll process completed"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 101
Rahul
150000.0
Output: Salary processed successfully for Rahul: 150000.0
Payroll process completed

Answer

```
import java.util.Scanner;
class InvalidSalaryException extends Exception

{
```

```
    public InvalidSalaryException(String message)
{

        super(message);


}

}
class Employee

{

    int empId;
    String empName;
    double salary;
    public Employee(int empId, String empName, double salary)

{


        this.empId = empId;
        this.empName = empName;
        this.salary = salary;


}
    public void processSalary() throws InvalidSalaryException

{


        if (salary < 0)

{


            throw new InvalidSalaryException("Salary cannot be negative");
```

```
} else if (salary > 200000)

{

        throw new InvalidSalaryException("Salary exceeds threshold limit");


} else

{

        System.out.println("Salary processed successfully for " + empName + ": "
+ salary);


}


}

}
class PayrollProcessor

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      int empId = sc.nextInt();
      sc.nextLine(); // consume newline
      String empName = sc.nextLine();
      double salary = sc.nextDouble();

      Employee emp = new Employee(empId, empName, salary);
```

```
        try
{

            emp.processSalary();

} catch (InvalidSalaryException e)

{

            System.out.println(e.getMessage());

} finally

{

            System.out.println("Payroll process completed");

}

}

}
```

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

4. Problem Statement

Daniel is developing a program to verify the age of users. He wants to ensure that the entered age is within a valid range. Write a program to help Daniel implement this age-checking feature using custom exceptions.

Daniel needs a program that takes an integer input representing a person's age.If the age is between 0 and 150 (inclusive), the program should print "Age is valid!".If the age is less than 0 or greater than 150, the program should throw a custom exception (InvalidAgeException) with the message "Invalid age. Please enter an age between 0 and 150."

Implement a custom exception, InvalidAgeException, to handle cases where the entered age does not meet the specified criteria.

Input Format

The input consists of an integer value 'n', representing the age.

Output Format

The output is displayed in the following format:

If the age is valid (between 0 and 150, inclusive), print

"Age is valid!".

If the age is invalid, print

"Error: Invalid age. Please enter an age between 0 and 150."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 45
Output: Age is valid!

Answer

```
import java.util.Scanner;
class InvalidAgeException extends Exception

{

    public InvalidAgeException(String message)
```

```
{
        super(message);

}
}
class AgeValidator
{
    public static void validateAge(int age) throws InvalidAgeException
{
        if (age < 0 || age > 150)
{
            throw new InvalidAgeException("Error: Invalid age. Please enter an age
between 0 and 150.");
} else
{
            System.out.println("Age is valid!");

}
}
    public static void main(String[] args)
```

```
{

        Scanner sc = new Scanner(System.in);
        int age = sc.nextInt();

        try

{


            validateAge(age);


} catch (InvalidAgeException e)
{


            System.out.println(e.getMessage());


}


}

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 8_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Theo is trying to update his payment information on a subscription-based streaming service. To proceed, the system requires Theo to provide a valid credit card number consisting of 16 digits. However, Theo wants to make sure that the credit card number he enters meets the specified criteria with proper exception handling.

The credit card number must consist of exactly 16 digits.If the entered credit card number does not meet the specified criteria, the program should throw a custom exception, InvalidCreditCardException, and provide Theo with specific error messages:If the length of the credit card number is not 16 digits, the exception message should be: "Invalid credit card number length."If the credit card number contains non-numeric characters, the exception message should be: "Invalid credit card number format."

Implement a custom exception, InvalidCreditCardException, to fulfill Theo's requirements and keep his payment information secure.

Input Format

The input consists of a string value 's', consisting of the 16-digit credit card number.

Output Format

The output is displayed in the following format:

If the entered credit card number is valid, the program should output a success message:

"Payment information updated successfully!"

If the entered credit card has more than 16 digits or less than 16 digits it displays

"Error: Invalid credit card number length."

If the entered 16-digit credit card has non-integers it displays

"Error: Invalid credit card number format."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1234567890123456
Output: Payment information updated successfully!

Answer

```
import java.util.Scanner;

// Custom exception
class InvalidCreditCardException extends Exception

{
```

```
    public InvalidCreditCardException(String message)
{

        super(message);


}

}

class CreditCardValidator
{

    public static void main(String[] args)

{


        Scanner sc = new Scanner(System.in);
        String s = sc.nextLine();

        try
{


            validateCreditCard(s);
            System.out.println("Payment information updated successfully!");


} catch (InvalidCreditCardException e)

{


            System.out.println("Error: " + e.getMessage());
```

```
}

}

   public static void validateCreditCard(String s) throws
InvalidCreditCardException

{


      if (s.length() != 16)

{


         throw new InvalidCreditCardException("Invalid credit card number
length.");


}
      if (!s.matches("\\d{16}"))

{


         throw new InvalidCreditCardException("Invalid credit card number
format.");


}


}

}
```

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

2. Problem Statement

A company is developing a user registration system that requires users to provide valid email addresses. The development team is implementing an EmailValidator program to ensure that the entered email addresses meet certain criteria using exception handling.

The email address must contain the "@" symbol.The email address must consist of a non-empty username(before "@" symbol) and a non-empty domain(after "'@" symbol).The domain part of the email address must contain at least one period (".").The email address must not contain leading or trailing spaces.

Implement a custom exception, InvalidEmailException, to fulfill the company's requirements and validate it according to the specified rules.

Input Format

The input consists of a string value 's', which represents the email address.

Output Format

The output is displayed in the following format:

If the entered email address is valid according to the specified rules, the program prints:

"Email address is valid!"

If the entered email address misses the username or domain part or misses "@" symbol or has two or more "@" symbols or misses '.' in the domain part it outputs:

"Error: Invalid email format."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: johndoe@example.com
Output: Email address is valid!

Answer

```
import java.util.Scanner;
class InvalidEmailException extends Exception
{
    public InvalidEmailException(String message)
{
        super(message);
}
}
class EmailValidator
{
    public static void validateEmail(String email) throws InvalidEmailException
{
        email = email.trim();
        if (!email.contains("@") || email.indexOf('@') != email.lastIndexOf('@'))
{
            throw new InvalidEmailException("Error: Invalid email format.");
}
        String[] parts = email.split("@");
```

```
        if (parts.length != 2 || parts[0].isEmpty() || parts[1].isEmpty() || !
parts[1].contains("."))
{

            throw new InvalidEmailException("Error: Invalid email format.");


}


}
    public static void main(String[] args)
{

        Scanner scanner = new Scanner(System.in);
        String email = scanner.nextLine();

        try

{


            validateEmail(email);
            System.out.println("Email address is valid!");


} catch (InvalidEmailException e)

{


            System.out.println(e.getMessage());


}
```

}
}

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

3. Problem Statement

Tim was tasked with creating a user profile system that validates the user's date of birth input. The system should throw a custom exception, InvalidDateOfBirthException, if the date is not in the specified format "dd-mm-yyyy" or if it represents an invalid calendar date.

The main method takes user input, validates the date of birth, and prints whether it is valid or not.

Input Format

The input consists of a string, representing the date of birth of the user.

Output Format

The output displays one of the following results:

If the entered date of birth is valid according to the specified format, the program prints:

"[Date] is a valid date of birth"

If the entered date of birth is not valid according to the specified format, the program prints:

"Invalid date: [Date]"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 01-01-2000

Output: 01-01-2000 is a valid date of birth

Answer

```
import java.util.Scanner;
import java.text.SimpleDateFormat;
import java.text.ParseException;

// Custom exception
class InvalidDateOfBirthException extends Exception

{


   public InvalidDateOfBirthException()

{


      super();


}

}

class DateOfBirthValidator

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      String dateInput = sc.nextLine();

      try

{
```

```
            validateDateOfBirth(dateInput);
            System.out.println(dateInput + " is a valid date of birth");

    } catch (InvalidDateOfBirthException e)

    {


            System.out.println("Invalid date: " + dateInput);


    }

    }

       public static void validateDateOfBirth(String date) throws
    InvalidDateOfBirthException

    {


          if (date.length() != 10)

    {


            throw new InvalidDateOfBirthException();


    }
          SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
          sdf.setLenient(false);
          try

    {


            sdf.parse(date);
```

```
} catch (ParseException ex)
{
        throw new InvalidDateOfBirthException();
}
}
}
```

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

4. Problem Statement

Hemanth is designing a banking system for XYZ Bank. The system should allow customers to perform deposit, withdrawal, and balance inquiry operations. Implement exception handling for scenarios involving invalid transaction amounts or insufficient funds.

Create two custom exception classes, InvalidAmountException and InsufficientFundsException, both extending the Exception class.Throw an InvalidAmountException with a message if the deposit amount is less than or equal to zero.Throw an InsufficientFundsException if the withdrawal amount is greater than the available balance.Deduct the withdrawal amount from the balance if the withdrawal is successful.

Assist Hemanth in designing the program.

Input Format

The first line of input consists of a double value B, representing the initial balance.

The second line consists of a double value D, representing the deposit amount.

The third line consists of a double value W, representing the withdrawal amount.

Output Format

If the withdrawal is successful, print the amount withdrawn and the current balance, rounded off to one decimal place.

If an InvalidAmountException occurs, print "Error: [D] is not valid".

If an InsufficientFundsException occurs, print "Error: Insufficient funds".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1050.1
270.2
150.3
Output: Amount Withdrawn: 150.3
Current Balance: 1170.0

Answer

```
import java.util.Scanner;

// Custom exception for invalid deposit amount
class InvalidAmountException extends Exception
{

   public InvalidAmountException(String message)

{

      super(message);

}
}
```

```
// Custom exception for insufficient funds
class InsufficientFundsException extends Exception

{


   public InsufficientFundsException(String message)

{


      super(message);


}

}

class BankSystem

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);

      double balance = Double.parseDouble(sc.nextLine());
      double deposit = Double.parseDouble(sc.nextLine());
      double withdraw = Double.parseDouble(sc.nextLine());

      try

{


         balance = deposit(balance, deposit);
         balance = withdraw(balance, withdraw);
```

```
            System.out.printf("Amount Withdrawn: %.1f\n", withdraw);
            System.out.printf("Current Balance: %.1f\n", balance);

    } catch (InvalidAmountException e)

    {


            System.out.printf("Error: %s is not valid\n", deposit);


    } catch (InsufficientFundsException e)

    {


            System.out.println("Error: Insufficient funds");


    }


    }

       public static double deposit(double balance, double depositAmt) throws
    InvalidAmountException

    {


          if (depositAmt <= 0)

    {


            throw new InvalidAmountException(Double.toString(depositAmt));


    }
          return balance + depositAmt;
```

```
}
   public static double withdraw(double balance, double withdrawAmt) throws
InsufficientFundsException

{


      if (withdrawAmt > balance)

{


         throw new InsufficientFundsException("Insufficient funds");


}
      return balance - withdrawAmt;


}

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 9_Q1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Bobby is tasked with processing a sequence of numbers from a monitoring system. He needs to extract a strictly increasing subsequence using an ArrayList. The program should dynamically add numbers to the ArrayList only if they are greater than the last number currently stored in the list. Bobby aims to efficiently utilize the dynamic resizing and indexing features of the ArrayList to solve this problem.

Help Bobby implement this solution.

Input Format

The first line of input consists of an integer N, representing the number of elements.

The second line consists of N space-separated integers, representing the elements.

Output Format

The output prints the list of integers in increasing sequence, ignoring out-of-order elements.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 7
3 5 9 1 11 7 13
Output: [3, 5, 9, 11, 13]

Answer

```
import java.util.*;

class Main

{

   public static void main(String args[])
{

      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();

      ArrayList<Integer> incList = new ArrayList<>();

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

{

         int num = sc.nextInt();
```

```
        if(incList.isEmpty() || num > incList.get(incList.size() - 1))
{

            incList.add(num);

}

}
        System.out.println(incList);

}
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 9_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Vikram loves listening to music and wants to create a simple playlist manager using Java Collections. The playlist supports the following operations:

"ADD <song>" Adds the song to the end of the playlist."REMOVE <song>" Removes the first occurrence of the song from the playlist. If the song is not found, do nothing."SHOW" Displays all songs in the playlist in order. If the playlist is empty, print "EMPTY"."NEXT" Moves to the next song in the playlist and prints its name. If the playlist is empty, print "EMPTY".

The playlist maintains a "current song" position that starts at the first song when it's added. The NEXT command moves to the next song and prints it, wrapping around to the first song after reaching the last song. When removing songs, the current position adjusts accordingly to maintain

proper navigation.

Help Vikram implement this playlist manager.

Input Format

The first line of the input consists of an integer n, the number of operations.

The next n lines, each containing a command:

- "ADD <song>"
- "REMOVE <song>"
- "SHOW"
- "NEXT"

Output Format

For each "SHOW" command, print the songs in order, separated by spaces.

For each "NEXT" command, print the next song in the playlist.

If no song exists, print "EMPTY".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
ADD song1
ADD song2
SHOW
NEXT
REMOVE song2
SHOW
NEXT

Output: song1 song2
song2
song1
song1

Answer

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

class Main

{


   public static void main(String args[])

{


      Scanner sc = new Scanner(System.in);
      int n = Integer.parseInt(sc.nextLine());

      LinkedList<String> playList = new LinkedList<>();
      int currentIndex = -1;

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

{


         String input = sc.nextLine();
         if(input.startsWith("ADD "))

{


            String song = input.substring(4);
            playList.add(song);
            if(currentIndex == -1)

{


               currentIndex = 0;


}
```

```
}
        else if(input.startsWith("REMOVE "))

{


            String song = input.substring(7);
            int index = playList.indexOf(song);
            if(index != -1)

{


                playList.remove(index);
                if(playList.isEmpty()) currentIndex = -1;


}else if(index < currentIndex)

{


                currentIndex--;


}else if(index == currentIndex)

{


                currentIndex = currentIndex % playList.size();


}


}else if(input.equals("SHOW"))

{
```

```
            if(playList.isEmpty())
    {

                System.out.println("EMPTY");

    }else

    {

                for(String song : playList)
    {

                    System.out.print(song + " ");

    }
                System.out.println();

    }
    }else if(input.equals("NEXT"))
    {

                if(playList.isEmpty())
    {

                System.out.println("EMPTY");
```

```
}else
{

        currentIndex = (currentIndex + 1) % playList.size();
        System.out.println(playList.get(currentIndex));


}

}

}

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 9_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Assist Pranitha in developing a program that takes an integer N as input, representing the number of names to be read. Then read N names and store them in an ArrayList. Finally, input a search string and output the frequency of that string in the list of names.

Note: Some parts of the code are provided as snippets, and you need to complete the remaining sections by writing the necessary code.

Input Format

The first line of input consists of an integer N, representing the number of names to be read.

The following N lines consist of N names, as a string.

The last line consists of a string, representing the name to be searched.

Output Format

The output prints a single integer, representing the frequency of the specified name in the given list.

If the specified name is not found, print 0.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
Alice
Bob
Ankit
Alice
Pranitha
Alice
Output: 2

Answer

```
import java.util.*;

class Main

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      int n = Integer.parseInt(sc.nextLine());

      ArrayList<String> names = new ArrayList<>();
```

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

            String name = sc.nextLine();
            names.add(name);


}

        String searchName = sc.nextLine();

        int count = 0;

        for(String name : names)

{


            if(name.equals(searchName))

{


                count++;


}


}
        System.out.println(count);


}

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 9_PAH

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Arun is building a task manager to keep track of tasks using a LinkedList. The task manager supports the following operations:

"ADD <task>" Adds the given task to the end of the list."REMOVE" Removes the first task from the list."SHOW" Displays all tasks in the list in order. If the list is empty, print "EMPTY".

Help Arun implement this functionality using a LinkedList.

Input Format

The first line of the input consists of an integer n, the number of operations.

The next n lines, each containing a command:

- "ADD <task>"
- "REMOVE"
- "SHOW"

Output Format

For each "SHOW" command, the output prints the tasks in order, separated by spaces.

If no tasks exist, print "EMPTY".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
ADD homework
ADD project
SHOW
REMOVE
SHOW
Output: homework project
project

Answer

```
import java.util.*;

public class Main

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      int n = Integer.parseInt(sc.nextLine());

      LinkedList<String> tasks = new LinkedList<>();
```

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

            String command = sc.nextLine();

            if (command.startsWith("ADD "))

{

                String task = command.substring(4);
                tasks.add(task);

} else if (command.equals("REMOVE"))

{

                if (!tasks.isEmpty())

{

                    tasks.removeFirst();

}

} else if (command.equals("SHOW"))

{

                if (tasks.isEmpty())

{
```

```
            System.out.println("EMPTY");
        } else
        {
            for (String task : tasks)
        {
              System.out.print(task + " ");
        }
            System.out.println();
        }
    }
}
```

Status : Correct *Marks : 10/10*

2. Problem Statement

Rekha is a teacher who wants to calculate the average of marks scored by her students in a test. She needs to store all the marks dynamically

because the number of students may vary each time. Using an ArrayList allows her to easily add any number of marks without worrying about the initial size.

Help her implement the task.

Input Format

The first line of input is an integer n, representing the number of students..

The second line of input consists of n double values, representing the marks of each student, separated by a space.

Output Format

The output prints: "Average of the list: " followed by the average value formatted to two decimal places.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1.0 2.0 3.0 4.0 5.0
Output: Average of the list: 3.00

Answer

```
import java.util.*;

public class Main

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      int n = Integer.parseInt(sc.nextLine().trim());
```

```
        String[] input = sc.nextLine().trim().split(" ");
        ArrayList<Double> marks = new ArrayList<>();
        for (int i = 0; i < n; i++)

{


            marks.add(Double.parseDouble(input[i]));


}
        double sum = 0.0;
        for (double mark : marks)

{


            sum += mark;


}
        System.out.printf("Average of the list: %.2f\n", sum / n);


}

}
```

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

3. Problem Statement

Aditi is analyzing stock market trends and wants to find the Next Greater Element (NGE) for each stock price in a list. The Next Greater Element for an element x in an array is the first element to the right that is greater than x. If no greater element exists, return -1 for that position.

Your task is to help Aditi by efficiently computing the Next Greater Element for each element in the given array using a Stack.

Example:

Input:

6

4 5 2 10 8 6

Output:

5 10 10 -1 -1 -1

Explanation:

For each element:

4 5 (next greater element)5 10 2 10 10 -1 (No greater element)8 -1 6

Input Format

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

The second line contains n space-separated integers arr[i], where arr[i] is the stock price on the i-th day.

Output Format

The output prints n space-separated integers representing the Next Greater Element for each element in the array.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6
4 5 2 10 8 6
Output: 5 10 10 -1 -1 -1

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 = Integer.parseInt(sc.nextLine().trim());
        String[] input = sc.nextLine().trim().split(" ");
        int[] arr = new int[n];
        int[] nge = new int[n];
        Stack<Integer> stack = new Stack<>();

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

            arr[i] = Integer.parseInt(input[i]);

}

        for (int i = n - 1; i >= 0; i--)
{

            while (!stack.isEmpty() && stack.peek() <= arr[i])
{

                stack.pop();

}
            nge[i] = stack.isEmpty() ? -1 : stack.peek();
            stack.push(arr[i]);
```

```
}
        for (int val : nge)
{

            System.out.print(val + " ");

}

}
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 9_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Sanjay is working on a program to merge two sorted linked lists into a single sorted list using Java's LinkedList class from the Collections framework. Given two sorted linked lists, he wants to merge them while maintaining the sorted order.

Write a Java program that:

Reads two sorted linked lists.Merges them into a single sorted linked list.Prints the merged list in ascending order.

Input Format

The first line contains an integer m (the size of the first linked list).

The second line contains m space-separated integers (sorted).

The third line contains an integer n (the size of the second linked list).

The fourth line contains n space-separated integers (sorted).

Output Format

The output prints the merged linked list as space-separated integers.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
5 10
3
1 3 8
Output: 1 3 5 8 10

Answer

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

public static void main(String[] args){


    Scanner sc = new Scanner(System.in);
    int m = Integer.parseInt(sc.nextLine().trim());
    LinkedList<Integer> list1 = new LinkedList<>();
    if (m > 0)

{


        String[] arr1 = sc.nextLine().trim().split(" ");
        for (int i = 0; i < m; i++)

{
```

```
            list1.add(Integer.parseInt(arr1[i]));

    }

    }
        int n = Integer.parseInt(sc.nextLine().trim());
        LinkedList<Integer> list2 = new LinkedList<>();
        if (n > 0)

    {

            String[] arr2 = sc.nextLine().trim().split(" ");
            for (int i = 0; i < n; i++)

    {

                list2.add(Integer.parseInt(arr2[i]));

    }

    }
        LinkedList<Integer> merged = new LinkedList<>();
        int i = 0, j = 0;
        while (i < list1.size() && j < list2.size())

    {

            if (list1.get(i) <= list2.get(j))

    {

                merged.add(list1.get(i));
                i++;
```

```
} else
{
            merged.add(list2.get(j));
            j++;

}
}
        while (i < list1.size())
{
            merged.add(list1.get(i));
            i++;

}
        while (j < list2.size())
{
            merged.add(list2.get(j));
            j++;

}
        for (int val : merged)
{
```

```
        System.out.print(val + " ");

    }


    }
    }
```

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

2. Problem Statement

Aarav is developing a music playlist application where users can manage their favorite songs. He wants to implement a feature that allows users to reorder the playlist by moving a song from one position to another.

You need to implement a function that performs the following operations using a LinkedList:

Add songs to the playlist in the given order.Move a song from a specified position to another position in the playlist.Print the final playlist after all operations.

Input Format

The first line of the input consists of an integer n representing the number of songs.

The next n lines, each containing a string representing a song name.

After the songs are given the next line contains an integer m, the number of move operations.

The next m lines, each containing two integers x and y representing the move operation where the song at position x (0-based index) should be moved to position y.

Output Format

The output prints the final playlist, each song on a new line.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
SongA
SongB
SongC
SongD
SongE
2
2 4
0 3
Output: SongB
SongD
SongE
SongA
SongC

Answer

```
import java.util.*;

class Main
{

   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      sc.nextLine();

      LinkedList<String> playList = new LinkedList<>();
      for(int i = 0; i < n; i++)
```

```
{

        String song = sc.nextLine();
        playList.add(song);


}

    int m = sc.nextInt();

    for(int i = 0; i < m; i++)
{

        int x = sc.nextInt();
        int y = sc.nextInt();

        String song = playList.remove(x);

        playList.add(y,song);


}

    for(String song : playList)
{


        System.out.println(song);


}


}
}
```

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

3. Problem Statement

Rahul is working on a list manipulation problem where he needs to reverse a specific subarray using a stack. Given an array and two indices l and r, he wants to reverse only the portion of the array from index l to r (both inclusive) while keeping the rest of the array unchanged.

Since Rahul wants to solve this problem efficiently, he decides to use a stack to reverse the subarray in O(r - l) time.

Your task is to help Rahul by implementing this functionality.

Input Format

The first line contains an integer n, the size of the array.

The second line contains n space-separated integers arr[i].

The third line contains two integers l and r, denoting the start and end indices of the subarray to reverse.

Note: The array follows 0-based indexing.

Output Format

The output prints the modified array after reversing the subarray between indices l and r.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6
1 2 3 4 5 6
1 4
Output: 1 5 4 3 2 6

Answer

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

   public static void main(String[] args)
{

      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      int arr[] = new int[n];

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

         arr[i] = sc.nextInt();

}

      int l = sc.nextInt();
      int r = sc.nextInt();

      Stack<Integer> stack = new Stack<>();

      for(int i = l; i <= r; i++)
{

         stack.push(arr[i]);

}
      for(int i = l; i <= r; i++)
{
```

```
            arr[i] = stack.pop();

        }

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

        {

            System.out.println(arr[i] + " ");

        }

    }
}
```

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

4. Problem Statement

Rahul, a stock trader, wants to analyze the stock prices of a company over several days. For each day, he wants to determine the stock span, which is the number of consecutive days (including the current day) where the stock price is less than or equal to the price on that day.

The stock span helps him understand how long a stock has been continuously increasing or staying the same. You need to help Rahul by computing the stock span for each day using a Stack data structure efficiently.

Example:

Input:

7

100 80 60 70 60 75 85

Output:

1 1 1 2 1 4 6

Explanation:

For each day:

Day 1: Price = 100 Span = 1 (Only this day)Day 2: Price = 80 Span = 1 (Only this day)Day 3: Price = 60 Span = 1 (Only this day)Day 4: Price = 70 Span = 2 (Includes today and previous day)Day 5: Price = 60 Span = 1 (Only this day)Day 6: Price = 75 Span = 4 (Includes today and previous three days)Day 7: Price = 85 Span = 6 (Includes today and previous five days)

Input Format

The first line contains an integer n, the number of days.

The second line contains n space-separated integers prices[i], where prices[i] represents the stock price on the i-th day.

Output Format

The output prints n space-separated integers representing the stock span for each day.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
100 80 60 70 60 75 85
Output: 1 1 1 2 1 4 6

Answer

```
import java.util.*;

public class Main
{
```

```
    public static void main(String[] args)
{

        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine().trim());
        String[] input = sc.nextLine().trim().split(" ");
        int[] prices = new int[n];
        int[] span = new int[n];
        Stack<Integer> stack = new Stack<>();

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

            prices[i] = Integer.parseInt(input[i]);


}

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

            while (!stack.isEmpty() && prices[stack.peek()] <= prices[i])
{

                stack.pop();


}
            span[i] = stack.isEmpty() ? (i + 1) : (i - stack.peek());
            stack.push(i);

}
```

```
        for (int s : span)
    {

            System.out.print(s + " ");

    }

    }
    }
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

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;
   String ownerName;
```

```
    String vehicleType;

    public Vehicle(String regNumber, String ownerName, String vehicleType)
{


        this.regNumber = regNumber;
        this.ownerName = ownerName;
        this.vehicleType = vehicleType;


}

    @Override
    public boolean equals(Object obj)
{


        if (this == obj) return true;
        if (!(obj instanceof Vehicle)) return false;
        Vehicle other = (Vehicle) obj;
        return this.regNumber.equals(other.regNumber);


}

    @Override
    public int hashCode()
{


        return regNumber.hashCode();


}

    @Override
    public String toString()
```

```
{

    return regNumber + " " + ownerName + " " + vehicleType;


}

}
class TollBoothTracker

{


   public static void main(String[] args)

{


    Scanner sc = new Scanner(System.in);
    int n = Integer.parseInt(sc.nextLine());

    Set<Vehicle> vehicleSet = new HashSet<>();

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

{


        String[] parts = sc.nextLine().split(" ");
        String regNumber = parts[0];
        String ownerName = parts[1];
        String vehicleType = parts[2];

        Vehicle vehicle = new Vehicle(regNumber, ownerName, vehicleType);
        vehicleSet.add(vehicle);


}

    for (Vehicle v : vehicleSet)
```

```
{
        System.out.println(v);

}

}

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

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

```
import java.util.*;
import java.text.DecimalFormat;
class FruitFestival

{

   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      Map<String, Double> fruitMap = new HashMap<>();
      DecimalFormat df = new DecimalFormat("0.00");

      while (true)

{


         String input = sc.nextLine();
```

```
        if (input.equalsIgnoreCase("done"))
{

            break;

}
        if (!input.contains(":") || input.indexOf(":") != input.lastIndexOf(":"))
{

            System.out.println("Invalid format");
            return;

}

        String[] parts = input.split(":");
        if (parts.length != 2)
{

            System.out.println("Invalid format");
            return;

}

        String fruitName = parts[0];
        String quantityStr = parts[1];

        try
{

            double quantity = Double.parseDouble(quantityStr);
            fruitMap.put(fruitName, quantity);
```

```
} catch (NumberFormatException e)

{

        System.out.println("Invalid input");
        return;


}

}
    double total = 0.0;
    for (double qty : fruitMap.values())

{

      total += qty;


}

    System.out.println(df.format(total));

}

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

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

```
import java.util.*;
class MessageAnalyzer

{


   public static void main(String[] args)

{
```

```
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());

        TreeMap<Character, Integer> frequencyMap = new TreeMap<>();

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

    {


            String line = sc.nextLine();
            for (char ch : line.toCharArray())

    {


                if (Character.isLetter(ch))

    {


                    frequencyMap.put(ch, frequencyMap.getOrDefault(ch, 0) + 1);


    }


    }

    }
        System.out.println("Character Frequency:");
        for (Map.Entry<Character, Integer> entry : frequencyMap.entrySet())

    {


            System.out.println(entry.getKey() + ": " + entry.getValue());


    }
```

```
}
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

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

```
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!");


}

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

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

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.*;
import java.text.DecimalFormat;

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 (Double.compare(this.gpa, other.gpa) != 0)

{


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


} else
{


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


}


}

    @Override
    public String toString()
```

```
{

        return id + " " + name + " " + String.format("%.2f", gpa);


}

    @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);


}

}
class StudentRecords

{


    public static void main(String[] args)
{
```

```
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());

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

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

    {


            String line = sc.nextLine();
            String[] parts = line.split(" ");
            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);


    }


    }

    }
```

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

2. 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 main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      int n = Integer.parseInt(sc.nextLine());

      TreeMap<String, String> events = new TreeMap<>();

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

{


         String line = sc.nextLine();
         String[] parts = line.split(" ", 2);
         String time = parts[0];
         String description = parts[1];

         // Only add if time is not already present
         events.putIfAbsent(time, description);


}

      System.out.println("Scheduled Events:");
      for (Map.Entry<String, String> entry : events.entrySet())

{
```

```
        System.out.println(entry.getKey() + " - " + entry.getValue());
    }

}

}
```

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

```
import java.util.*;

public class Main

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      int N = sc.nextInt();
      String s = sc.next();
      HashMap<Character, Integer> map = new HashMap<>();
      for (char c : s.toCharArray())

{


         map.put(c, map.getOrDefault(c, 0) + 1);


}
      char result = '-';
      for (char c : s.toCharArray())

{


         if (map.get(c) == 1)

{


            result = c;
```

```
            break;

        }


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


    }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

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.*;
```

```
import java.util.*;

class Book

{


   private int isbn;
   private String title;
   private String author;

   public Book(int isbn, String title, String author)

{


      this.isbn = isbn;
      this.title = title;
      this.author = author;


}

   public int getIsbn()

{


      return isbn;


}

   public String toString()

{


      return "ISBN: " + isbn + ", Title: " + title + ", Author: " + author;

}
```

```
    public boolean equals(Object o)
    {

        if (this == o) return true;
        if (!(o instanceof Book)) return false;
        Book b = (Book) o;
        return isbn == b.isbn;


    }
    public int hashCode()
    {

        return Objects.hash(isbn);


    }

    }
    class Library
    {

       private 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(b -> b.getIsbn() == 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

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

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

import java.util.HashMap;

class MajorityElementFinder

{


   public static int findMajorityElement(int[] arr)

{


      HashMap<Integer, Integer> voteCount = new HashMap<>();
      int n = arr.length;
      int threshold = n / 2;

      for (int vote : arr)

{


         voteCount.put(vote, voteCount.getOrDefault(vote, 0) + 1);
         if (voteCount.get(vote) > threshold)

{


            return vote;


}


}

      return -1;
```

```
}
}
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();
    }
}
```

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

3. 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;


}

    @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 other = (Student) obj;
        return rollNo == other.rollNo;


}
```

```
    @Override
    public int hashCode()
{

        return Objects.hash(rollNo);


}

}

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++)

{


            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();
        // Find student and increase attendance
        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***

4. 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.*;

import java.util.HashMap;
class ScoreTracker
{


   HashMap<String, Integer> scoreMap = new HashMap<>();
   boolean invalidInput = false;
   boolean invalidFormat = false;

   public boolean processInput(String input)

{


      // Check if exactly one ':' exists
      if (input.chars().filter(ch -> ch == ':').count() != 1)

{


         invalidFormat = true;
         System.out.println("Invalid format");
         return false;


}
      // Remove ':' for special char check
      String temp = input.replace(":", "");
```

```
        if (!temp.matches("[a-zA-Z0-9]+"))
    {

            invalidFormat = true;
            System.out.println("Invalid format");
            return false;


    }
        String[] parts = input.split(":");
        if (parts.length != 2)
    {

            invalidFormat = true;
            System.out.println("Invalid format");
            return false;


    }
        String player = parts[0].trim();
        String scoreStr = parts[1].trim();

        if (player.length() < 1 || player.length() > 20)
    {

            invalidInput = true;
            System.out.println("Invalid input");
            return false;


    }
        int score = 0;
        try
    {
```

```
            score = Integer.parseInt(scoreStr);

        } catch (NumberFormatException e)

        {


            invalidInput = true;
            System.out.println("Invalid input");
            return false;

        }
            if (score < 1 || score > 100)

        {


            invalidInput = true;
            System.out.println("Invalid input");
            return false;


        }
            scoreMap.put(player, score);
            return true;


        }

       public String findTopPlayer()

        {


            String topPlayer = "";
            int maxScore = Integer.MIN_VALUE;
            for (String name : scoreMap.keySet())

        {
```

```
        int val = scoreMap.get(name);
        if (val > maxScore)

{


            maxScore = val;
            topPlayer = name;


}


}
    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**

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 11

Attempt : 1
Total Mark : 20
Marks Obtained : 10

Section 1 : Project

1. Problem Statement

Create a JDBC-based School Management System that handles runtime input to manage student records. The system should allow users to:

Add a new student (student ID, name, grade level, GPA).

Update a student's GPA, ensuring the GPA value is within the valid range (0.0 - 4.0).

View a specific student’s record by student ID.

Display all students in the database.

Exit the application.

The system should connect to a MySQL database using the following default credentials:

DB URL: jdbc:mysql://localhost/ri_db

USER: test

PWD: test123

The students table has already been created with the following structure:

Table Name: students

Input Format

The first line of input consists of an integer choice, representing the operation to be performed:

(1 for Add Student, 2 for Update GPA, 3 for View Student Record, 4 for Display All Students, 5 for Exit)

For choice 1 (Add Student):

- The second line consists of an integer student_id.
- The third line consists of a string name.
- The fourth line consists of a string grade_level.
- The fifth line consists of a double gpa (must be between 0.0 and 4.0).

For choice 2 (Update GPA):

- The second line consists of an integer student_id.
- The third line consists of a double new_gpa (must be between 0.0 and 4.0).

For choice 3 (View Student Record):

- The second line consists of an integer student_id.

For choice 4 (Display All Students):

- No additional inputs are required.

For choice 5 (Exit):

- No additional inputs are required.

Output Format

The output displays:

For choice 1 (Add Student):

- Print "Student added successfully" if the student was added.
- Print "Failed to add student." if the insertion failed.

For choice 2 (Update GPA):

- Print "GPA updated successfully" if the GPA update was successful.
- Print "Student not found." if the specified student ID does not exist.
- Print "GPA must be between 0.0 and 4.0." if the provided GPA is out of the valid range.

For choice 3 (View Student Record):

- Display the student details in the format:
- ID: [student_id] | Name: [name] | Grade Level: [grade_level] | GPA: [gpa]
- Print "Student not found." if the specified student ID does not exist.

For choice 4 (Display All Students):

- Display each student on a new line in the format:
- ID | Name | Grade Level | GPA
- If there are no records, print nothing (or handle with an appropriate message if desired).

For choice 5 (Exit):

- Print "Exiting School Management System."

For invalid input:

- Print "Invalid choice. Please try again."

Sample Test Case

Input: 1
101
Alice Johnson
10

3.8
5
Output: Student added successfully
Exiting School Management System.

Answer

```
import java.sql.*;
import java.util.Scanner;

class SchoolManagementSystem {
   public static void main(String[] args) {
      try (Connection conn = DriverManager.getConnection("jdbc:mysql://
localhost/ri_db", "test", "test123");
       Scanner scanner = new Scanner(System.in)) {

        boolean running = true;

        while (running) {

          int choice = scanner.nextInt();

          switch (choice) {
            case 1:
              addStudent(conn, scanner);
              break;
            case 2:
              updateGrades(conn, scanner);
              break;
            case 3:
              viewStudentRecord(conn, scanner);
              break;
            case 4:
              displayAllStudents(conn);
              break;
            case 5:
              System.out.println("Exiting School Management System.");
              running = false;
              break;
            default:
              System.out.println("Invalid choice. Please try again.");
          }
        }
```

```
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    public static void addStudent(Connection conn, Scanner scanner){



        int studentId = scanner.nextInt();
        scanner.nextLine(); // consume newline
        String name = scanner.nextLine();
        String gradeLevel = scanner.nextLine();
        double gpa = scanner.nextDouble();
        scanner.nextLine(); // consume newline

        if (gpa < 0.0 || gpa > 4.0)

{


            System.out.println("GPA must be between 0.0 and 4.0.");
            return;


}
        String sql = "INSERT INTO students (student_id, name, grade_level, gpa)
VALUES (?, ?, ?, ?)";
        try (PreparedStatement pstmt = conn.prepareStatement(sql))

{


            pstmt.setInt(1, studentId);
            pstmt.setString(2, name);
            pstmt.setString(3, gradeLevel);
            pstmt.setDouble(4, gpa);
            int rows = pstmt.executeUpdate();
            if (rows > 0)

{
```

```
            System.out.println("Student added successfully");

        } else
        {

            System.out.println("Failed to add student.");

        }
        } catch (SQLException e)
        {

            System.out.println("Failed to add student.");

        }

        }
    public static void updateGrades(Connection conn, Scanner scanner)
    {

        int studentId = scanner.nextInt();
        double newGpa = scanner.nextDouble();
        scanner.nextLine(); // consume newline

        if (newGpa < 0.0 || newGpa > 4.0)
    {
```

```
            System.out.println("GPA must be between 0.0 and 4.0.");
            return;


}
        try (PreparedStatement pstmt = conn.prepareStatement("UPDATE students
SET gpa = ? WHERE student_id = ?"))

{


            pstmt.setDouble(1, newGpa);
            pstmt.setInt(2, studentId);
            int rows = pstmt.executeUpdate();
            if (rows > 0)

{


               System.out.println("GPA updated successfully");


} else

{


               System.out.println("Student not found.");


}


} catch (SQLException e)

{


            System.out.println("Student not found.");


}
```

```
}
   public static void viewStudentRecord(Connection conn, Scanner scanner)
{

      int studentId = scanner.nextInt();
      scanner.nextLine(); // consume newline
      try (PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM
students WHERE student_id = ?"))
{

         pstmt.setInt(1, studentId);
         ResultSet rs = pstmt.executeQuery();
         if (rs.next())
{

            System.out.printf("ID: %d | Name: %s | Grade Level: %s | GPA: %.2f%n",
                  rs.getInt("student_id"),
                  rs.getString("name"),
                  rs.getString("grade_level"),
                  rs.getDouble("gpa"));

} else
{

            System.out.println("Student not found.");

}
```

```
} catch (SQLException e)
{

        System.out.println("Student not found.");


}


}

   public static void displayAllStudents(Connection conn)
{

      try (Statement stmt = conn.createStatement())

{


        ResultSet rs = stmt.executeQuery("SELECT * FROM students");
        boolean headerPrinted = false;
        while (rs.next())

{


          if (!headerPrinted)

{


            System.out.println("ID | Name | Grade Level | GPA");
            headerPrinted = true;


}
          System.out.printf("%d | %s | %s | %.2f%n",
              rs.getInt("student_id"),
```

```
                rs.getString("name"),
                rs.getString("grade_level"),
                rs.getDouble("gpa"));


}


} catch (SQLException e)

{


    // Silent fail or print nothing


}
}


}
```

Status : Wrong ***Marks : 0/10***

2. Problem Statement

In ABC Corporation, employee records are stored in a database.

To efficiently manage employee details using Java and JDBC, you are tasked with building an Employee Management System that supports the following functionalities:

Adding a new employee

Updating an employee's salary

Viewing an employee's details

Displaying all employees

You are given two files:

File 1: Employee.java (POJO Class)

This class represents the Employee entity.

An Employee contains the following details:

Field Description

employeeId Unique Employee ID (Integer)

name Employee Name (String)

department Employee Department (String)

salary Employee Salary (Double)

Students must write code in the marked area:

```
class Employee {
    private int employeeId;
    private String name;
    private String department;
    private double salary;

    public Employee() {}

    public Employee(int employeeId, String name, String department, double salary) {
        // write your code here
    }

    // Include getters and setters
}
```

Expected in this part:

Assign parameter values to instance variables inside the constructor.

Add getters and setters for all attributes.

File 2: EmployeeDAO.java (Data Access Layer)

This class handles all database operations using JDBC.

Students must complete the missing JDBC logic in the following methods:

```
class EmployeeDAO {

    public void addEmployee(Connection conn, Employee employee) throws
SQLException {
        // write your code here
    }

    public void updateSalary(Connection conn, int employeeId, double
newSalary) throws SQLException {
        // write your code here
    }

    public void deleteEmployee(Connection conn, int employeeId) throws
SQLException {
        // write your code here
    }

    public Employee viewEmployeeRecord(Connection conn, int employeeId)
throws SQLException {
        // write your code here
    }

    public List<Employee> displayAllEmployees(Connection conn) throws
SQLException {
        // write your code here
    }

    private Employee mapToEmployee(ResultSet rs) throws SQLException {
```

```
        return new Employee(
            // write your code here
        );
    }
}
```

Expected in this part:

Write SQL queries for INSERT, UPDATE, DELETE, SELECT.

Execute queries using PreparedStatement or Statement.

Map ResultSet rows to Employee objects using mapToEmployee().

Return a List<Employee> where required.

The system should connect to a MySQL database using the following default credentials:

DB URL: jdbc:mysql://localhost/ri_dbUsername: testPassword: test123

The employees table has already been created with the following structure:

Input Format

The first line of input consists of an integer choice, representing the operation to be performed:

(1 for Add Employee, 2 for Update Salary, 3 for View Employee Record, 4 for Display All Employees, 5 for Exit)

For choice 1 (Add Employee):

1. The second line consists of an integer employee_id.
2. The third line consists of a string name.
3. The fourth line consists of a string department.
4. The fifth line consists of a double salary (must be at least 30000).

For choice 2 (Update Salary):

1. The second line consists of an integer employee_id.
2. The third line consists of a double new_salary (must be at least 30000).

For choice 3 (View Employee Record):

1. The second line consists of an integer employee_id.

For choice 4 (Display All Employees).

For choice 5 (Exit).

Output Format

For choice 1 (Add Employee),

1. Print "Employee added successfully" if the employee was added.

For choice 2 (Update Salary),

1. Print "Salary updated successfully" if the salary update was successful.
2. Print "Employee not found." if the specified employee ID does not exist.
3. Print "Salary must be at least 30000." if the provided salary is below the minimum.

For choice 3 (View Employee Record),

1. Display the employee details in the format:
2. ID: [employee_id] | Name: [name] | Department: [department] | Salary: [salary]
3. Print "Employee not found." if the specified employee ID does not exist.

For choice 4 (Display All Employees),

1. Display each employee on a new line in the format:
2. ID | Name | Department | Salary

For choice 5 (Exit),

1. Print "Exiting Employee Management System."

For invalid input:

1. Print "Invalid choice. Please try again."

Sample Test Case

Input: 1
101
Alice Johnson
Engineering
31000.75
4
6
5
Output: Employee added successfully
ID | Name | Department | Salary
101 | Alice Johnson | Engineering | 31000.75
Invalid choice. Please try again.
Exiting Employee Management System.

Answer

```
import java.sql.*;
import java.util.Scanner;

class Employee{


    private int employeeId;
    private String name;
    private String department;
```

```
    private double salary;

    // Constructor
    public Employee(int employeeId, String name, String department, double
salary)

{


        this.employeeId = employeeId;
        this.name = name;
        this.department = department;
        this.salary = salary;


}

    // Getters and Setters
    public int getEmployeeId()

{

 return employeeId;

}
    public void setEmployeeId(int employeeId)

{

 this.employeeId = employeeId;

}

    public String getName()

{

 return name;

}
    public void setName(String name)
```

```
{
 this.name = name;
}
    public String getDepartment()
{
 return department;
}
    public void setDepartment(String department)
{
 this.department = department;
}
    public double getSalary()
{
 return salary;
}
    public void setSalary(double salary)
{
 this.salary = salary;
}
}

class EmployeeManagementSystem
{
```

```
    // Add Employee
    public static void addEmployee(Connection conn, Scanner scanner)

{


        int employeeId = scanner.nextInt();
        scanner.nextLine();  // Consume newline
        String name = scanner.nextLine();
        String department = scanner.nextLine();
        double salary = scanner.nextDouble();

        if (salary < 30000)

{


            System.out.println("Salary must be at least 30000.");
            return;


}

        // Create an Employee POJO object
        Employee employee = new Employee(employeeId, name, department,
salary);

        String insertQuery = "INSERT INTO employees (employee_id, name,
department, salary) VALUES (?, ?, ?, ?)";
        try (PreparedStatement stmt = conn.prepareStatement(insertQuery))

{


            stmt.setInt(1, employee.getEmployeeId());
            stmt.setString(2, employee.getName());
            stmt.setString(3, employee.getDepartment());
            stmt.setDouble(4, employee.getSalary());
```

```
            int rowsInserted = stmt.executeUpdate();
            System.out.println(rowsInserted > 0 ? "Employee added successfully" :
"Failed to add employee.");


} catch (SQLException e)

{


            System.out.println("Error adding employee: " + e.getMessage());


}


}

   // Update Salary
   public static void updateSalary(Connection conn, Scanner scanner)

{


      int employeeId = scanner.nextInt();
      double newSalary = scanner.nextDouble();

      if (newSalary < 30000)

{


            System.out.println("Salary must be at least 30000.");
            return;


}

      String updateQuery = "UPDATE employees SET salary = ? WHERE
employee_id = ?";
      try (PreparedStatement stmt = conn.prepareStatement(updateQuery))
```

```
{


        stmt.setDouble(1, newSalary);
        stmt.setInt(2, employeeId);

        int rowsUpdated = stmt.executeUpdate();
        System.out.println(rowsUpdated > 0 ? "Salary updated successfully" :
"Employee not found.");


} catch (SQLException e)

{


        System.out.println("Error updating salary: " + e.getMessage());


}


}

   // View Employee Record
   public static void viewEmployeeRecord(Connection conn, Scanner scanner)

{


      int employeeId = scanner.nextInt();
      String selectQuery = "SELECT * FROM employees WHERE employee_id = ?";

      try (PreparedStatement stmt = conn.prepareStatement(selectQuery))

{


        stmt.setInt(1, employeeId);
        ResultSet rs = stmt.executeQuery();

        if (rs.next())
```

```
{
            Employee employee = new Employee(
                rs.getInt("employee_id"),
                rs.getString("name"),
                rs.getString("department"),
                rs.getDouble("salary")
            );
            System.out.printf("ID: %d | Name: %s | Department: %s | Salary: %.2f%n",
                employee.getEmployeeId(),
                employee.getName(),
                employee.getDepartment(),
                employee.getSalary());

} else

{

            System.out.println("Employee not found.");

}

} catch (SQLException e)

{

        System.out.println("Error retrieving employee record: " + e.getMessage());

}

}
   // Display All Employees
```

```
    public static void displayAllEmployees(Connection conn)
{

        String displayQuery = "SELECT * FROM employees";

        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(displayQuery))

{


            System.out.println("ID | Name | Department | Salary");
            while (rs.next())

{


                Employee employee = new Employee(
                        rs.getInt("employee_id"),
                        rs.getString("name"),
                        rs.getString("department"),
                        rs.getDouble("salary")
                );
                System.out.printf("%d | %s | %s | %.2f%n",
                        employee.getEmployeeId(),
                        employee.getName(),
                        employee.getDepartment(),
                        employee.getSalary());


}


} catch (SQLException e)

{


            System.out.println("Error displaying employees: " + e.getMessage());
```

```
}


}
   public static void main(String[] args) {
      String url = "jdbc:mysql://localhost/ri_db";
      String username = "test";
      String password = "test123";

      try (Connection conn = DriverManager.getConnection(url, username,
password);
          Scanner scanner = new Scanner(System.in)) {

         int choice;
         do {
            choice = scanner.nextInt();

            switch (choice) {
               case 1 -> addEmployee(conn, scanner);
               case 2 -> updateSalary(conn, scanner);
               case 3 -> viewEmployeeRecord(conn, scanner);
               case 4 -> displayAllEmployees(conn);
               case 5 -> System.out.println("Exiting Employee Management
System.");
               default -> System.out.println("Invalid choice. Please try again.");
            }

         } while (choice != 5);

      } catch (SQLException e) {
         System.out.println("Database Error: " + e.getMessage());
      }
   }
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 12_Q1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Sabrina is working on a project that involves analyzing a set of numbers. In her exploration, she encounters scenarios where extracting even numbers and finding their sum is essential.

Create a program that calculates the sum of even numbers from a given array of integers using a lambda expression.

Input Format

The first line of input consists of an integer N, representing the size of the array.

The second line consists of N space-separated integers, representing the elements of the array.

Output Format

The output prints the sum of the even integers from the array.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
29 37 45

Output: 0

Answer

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

public class Main

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      int[] arr = new int[n];
      for (int i = 0; i < n; i++) arr[i] = sc.nextInt();
      int sum = Arrays.stream(arr).filter(x -> x % 2 == 0).sum();
      System.out.println(sum);


}

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 12_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Alex is learning about Java's functional interfaces and lambda expressions.

He wants to write a simple program that prints the square of each number in an array using a predefined functional interface.

Help Alex complete this task using the Consumer functional interface.

Input Format

- The first line contains an integer N, the number of elements in the array.
- The second line contains N space-separated integers.

Output Format

- Print the squares of all elements in the array, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
1 2 3 4
Output: 1 4 9 16

Answer

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

public class Main

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      int[] arr = new int[n];
      for (int i = 0; i < n; i++) arr[i] = sc.nextInt();
      Consumer<Integer> square = x -> System.out.print((x * x) + " ");
      for (int num : arr) square.accept(num);


}

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 12_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In the mystical realm of programming, there exists a magical incantation to reveal hidden words.

Elara, the skilled enchantress, wishes to summon a word using her spell and then reverse its characters to uncover its enchanted reflection.

Write a program that uses the predefined functional interface Supplier<String> and a lambda expression to:

Supply (generate) a string, and

Display its reversed form.

Input Format

No input is required from the user.

The string must be supplied internally using a Supplier<String>.

Output Format

Print the reversed version of the supplied string.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Wizard!!
Output: !!draziW

Answer

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

public class Main

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      String input = sc.nextLine();
      Supplier<String> supplier = () -> new StringBuilder(input).reverse().toString();
      System.out.println(supplier.get());


}

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 12_Q4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Abi is working on a text analysis project where she needs to categorize words based on their length.

Words that have three or fewer characters are considered “Short”, while

words with more than three characters are classified as “Long.”

Write a Java program that takes a sentence as input, analyzes each word, and prints a list showing whether each word is “Short” or “Long.”

Use the predefined functional interface Function<String, String> along with a lambda expression for categorization.

Input Format

A single line containing a sentence (words separated by spaces).

Output Format

- A single line with each word categorized as "Short" or "Long", separated by spaces.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: I love my cat
Output: Short Long Short Short

Answer

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

public class Main

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      String sentence = sc.nextLine();
      String[] words = sentence.split(" ");
      Function<String, String> categorize = w -> w.length() <= 3 ? "Short" : "Long";
      for (String word : words)

{


         System.out.print(categorize.apply(word) + " ");


}
```

```
}
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_Week 12_Java_Lamba Expressions_PAH

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : COD

1. Problem Statement

Sneha is developing a feature for an e-commerce application that helps display product details after applying a seasonal discount.

She decides to use lambda expressions with the Consumer functional interface to print each product's name, original price, and discounted price neatly.

The program should:

Accept a list of product names and their prices.Apply a 15% discount on all products.Use a Consumer lambda expression to display the details in a formatted manner.

Input Format

The first line of input consists of an integer n, representing the number of products.

The next n lines each contain a String (product name) and a double (price) separated by a space.

Output Format

For each product, print the details in the format:

Product: <name>, Original Price: <price>, Discounted Price: <discounted price>

If there are no products, print:

No products available

Sample Test Case

Input: 1
Phone 60000
Output: Product: Phone, Original Price: 60000.0, Discounted Price: 51000.0

Answer

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

class Product

{


   String name;
   double price;
   Product(String name, double price)

{


      this.name = name;
      this.price = price;


}
```

```
}
public class Main
{
   public static void main(String[] args)
{
      Scanner sc = new Scanner(System.in);
      if (!sc.hasNextInt())
{
         System.out.println("No products available");
         return;
}
      int n = sc.nextInt();
      sc.nextLine();
      if (n <= 0)
{
         System.out.println("No products available");
         return;
}
      List<Product> products = new ArrayList<>();
      for (int i = 0; i < n; i++)
{
```

```
        String line = sc.nextLine().trim();
        while (line.isEmpty() && sc.hasNextLine()) line = sc.nextLine().trim();
        int lastSpace = line.lastIndexOf(' ');
        if (lastSpace == -1)

{


            products.add(new Product(line, 0.0));


} else

{


            String name = line.substring(0, lastSpace);
            double price = Double.parseDouble(line.substring(lastSpace + 1));
            products.add(new Product(name, price));


}


}
    Consumer<Product> display = p ->

{


        double discounted = p.price * 0.85;
        System.out.println("Product: " + p.name + ", Original Price: " +
String.format("%.1f", p.price) + ", Discounted Price: " + String.format("%.1f",
discounted));


};
    products.forEach(display);


}
```

}

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

2. Problem Statement

Emily, an analyst at a data processing firm, is tasked with cleaning up datasets to remove duplicate values from lists of integers.

Create a Java program that allows Emily to input a series of integers, with the program then utilizing a lambda expression to efficiently remove any duplicates.

Input Format

The first line of input consists of an integer N, representing the size of the array.

The second line consists of N space-separated integers, each denoting an array element.

Output Format

The output prints the array elements after removing the duplicates inside the square bracket separated by a comma and space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 15
1 2 3 4 3 2 1 2 3 4 4 4 5 5 6
Output: [1, 2, 3, 4, 5, 6]

Answer

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

```
public class Main
{

   public static void main(String[] args)
{

      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      List<Integer> list = new ArrayList<>();
      for (int i = 0; i < n; i++) list.add(sc.nextInt());
      Function<List<Integer>, List<Integer>> removeDuplicates = l ->
l.stream().distinct().collect(Collectors.toList());
      List<Integer> result = removeDuplicates.apply(list);
      System.out.println(result);


}

}
```

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

3. Problem Statement

Rishi is working as an HR analyst in a software company. He wants to filter a list of employees based on their salary using modern Java techniques. He has a list of employee names and salaries and wants to use lambda expressions to filter those who earn more than a specific threshold.

Implement a program using lambda expressions and functional interfaces to print the names of employees whose salary is greater than or equal to 50,000.

Input Format

The first line of input consists of an integer n, representing the number of employees.

The next n lines. Each line contains a String (employee name) and an int (salary).

Output Format

The output prints the names of employees whose salary is greater than or equal to 50000, each on a new line.

If no employee found with salary greater than 50000, print: No employee found with salary >= 50000

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
Amit 45000
Sneha 50000
Ravi 60000
Priya 30000

Output: Sneha
Ravi

Answer

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

class Employee

{


   String name;
   int salary;
   Employee(String name, int salary)

{
```

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


}

}

public class Main

{


   public static void main(String[] args)

{


        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        List<Employee> employees = new ArrayList<>();
        for (int i = 0; i < n; i++)

{


            String name = sc.next();
            int salary = sc.nextInt();
            employees.add(new Employee(name, salary));


}
        Predicate<Employee> highSalary = e -> e.salary >= 50000;
        List<Employee> filtered =
employees.stream().filter(highSalary).collect(Collectors.toList());
        if (filtered.isEmpty())

{


            System.out.println("No employee found with salary >= 50000");
```

```
} else
{
        filtered.forEach(e -> System.out.println(e.name));
}
}
}
```

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

4. Problem Statement

Aditya is developing a reading app that recommends books to users based on a predefined list.

Each time a user opens the app, it should supply the next book title in the list, one at a time, using a lambda expression and the Supplier functional interface.

When all books have been recommended, the list should start again from the beginning.

Input Format

The first line contains an integer n — the total number of available book titles.

The next n lines each contain a book title (a string).

The next line contains an integer m — the number of times users open the app (i.e., the number of recommendations to be made).

Output Format

Print the supplied book title for each recommendation, one per line.

If m > n, repeat the list from the start.

Sample Test Case

Input: 3
The Alchemist
Atomic Habits
Ikigai
5
Output: The Alchemist
Atomic Habits
Ikigai
The Alchemist
Atomic Habits

Answer

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

public class Main

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      sc.nextLine();
      List<String> books = new ArrayList<>();
      for (int i = 0; i < n; i++) books.add(sc.nextLine());
      int m = sc.nextInt();
      final int[] index =

{

0

};
```

```
        Supplier<String> supplier = () ->
{

            String book = books.get(index[0]);
            index[0] = (index[0] + 1) % books.size();
            return book;


};
        for (int i = 0; i < m; i++) System.out.println(supplier.get());


}
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_Week 12_Java_Lamba Expressions_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Nethra is a researcher working on a project that involves analyzing experimental data. As part of her analysis, she needs to determine whether a given word is a palindrome or not.

Create a Java program that allows Nethra to input a word, and then check and display whether the entered word is a palindrome. Use lambda expressions to perform the palindrome check.

Input Format

The first line of input consists of a word.

Output Format

The output prints whether the given word is a palindrome or not in the following format:

"<input> is palindrome" or "<input> is not palindrome".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: malayalam
Output: malayalam is palindrome

Answer

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

public class Main

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      String word = sc.nextLine();
      Predicate<String> isPalindrome = s -> s.equalsIgnoreCase(new
StringBuilder(s).reverse().toString());
      if (isPalindrome.test(word))
         System.out.println(word + " is palindrome");
      else
         System.out.println(word + " is not palindrome");


}

}
```

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

2. Problem Statement

Problem Statement

Sophia, a data analyst, is studying experimental results collected from various lab sensors. Each sensor provides a list of numeric readings, and Sophia wants to calculate the average of these readings to analyze consistency.

She decides to use lambda expressions and the Function functional interface to compute the average of all the recorded values efficiently.

Your Task

Write a Java program that:

Reads the total number of measurements.Reads all the measurement values as doubles.Uses a Function<double[], Double> lambda expression to calculate the average value.Displays the final average, formatted to two decimal places.

Input Format

The first line of input consists of an integer N, representing the number of measurements.

The second line contains N space-separated double values.

Output Format

Print the average of the entered values, rounded to two decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6
2.2 1.2 5.4 4.6 2.9 55.7
Output: 12.00

Answer

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

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

public class Main

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      double[] readings = new double[n];
      for (int i = 0; i < n; i++) readings[i] = sc.nextDouble();
      Function<double[], Double> average = arr ->

{


         double sum = 0;
         for (double val : arr) sum += val;
         return sum / arr.length;


};
      System.out.printf("%.2f", average.apply(readings));


}

}
```

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

3. Problem Statement

Riya is developing a college admission system that assigns unique roll numbers to each newly admitted student.

Each roll number should follow this fixed format:

<DEPT>-<YEAR>-<4-digit-sequence>

where:

<DEPT> is the department code (in uppercase, e.g., CSE, ECE, MECH).<YEAR> is the admission year (e.g., 2025).<4-digit-sequence> starts from a given number and increases sequentially for each student.Write a Java program using a Supplier<String> lambda to generate and print the roll numbers for n students.

Input Format

First line: integer n — number of roll numbers to generate

Second line: string DEPT — department code (uppercase letters only)

Third line: integer YEAR — admission year

Fourth line: integer start — starting sequence number (0 ≤ start ≤ 9999)

Output Format

Print n roll numbers, one per line, in the required format

Sequence must be zero-padded to 4 digits

If sequence exceeds 9999, wrap around to 0000

Sample Test Case

Input: 5
CSE
2025
98

Output: CSE-2025-0098
CSE-2025-0099
CSE-2025-0100
CSE-2025-0101
CSE-2025-0102

Answer

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

```
public class Main
{

   public static void main(String[] args)
{

      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      String dept = sc.next();
      int year = sc.nextInt();
      int start = sc.nextInt();
      final int[] seq =
{
start
};
      Supplier<String> rollSupplier = () ->
{
         String roll = String.format("%s-%d-%04d", dept, year, seq[0]);
         seq[0] = (seq[0] + 1) % 10000;
         return roll;
};
      for (int i = 0; i < n; i++)
{
         System.out.println(rollSupplier.get());
```

```
}

}

}
```

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

4. Problem Statement

A company named TechNova is collecting feedback from its customers. Each customer gives a feedback score (an integer between 1 and 10) along with their name.

The company wants to:

Display each customer's name along with their feedback in a formatted way using a lambda expression and a Consumer functional interface.After displaying all feedbacks, calculate and display the average feedback score.You need to implement this functionality using Java lambda expressions and streams, emphasizing the Consumer interface for displaying formatted output.

Input Format

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

The next n lines each contain a String (customer name) followed by an int (feedback score).

Output Format

- Each line prints a customer's name and feedback in the format:
- Customer: <name>, Feedback Score: <score>

- After all customers are displayed, print the average feedback as:
- Average Feedback: <average_value>

(Average should be displayed up to two decimal places.)

Sample Test Case

Input: 3
Ravi 7
Ananya 9
Kiran 8

Output: Customer: Ravi, Feedback Score: 7
Customer: Ananya, Feedback Score: 9
Customer: Kiran, Feedback Score: 8
Average Feedback: 8.00

Answer

```
import java.util.*;
import java.util.function.*;
import java.util.stream.*;

class Customer

{


   String name;
   int feedback;
   Customer(String name, int feedback)

{


      this.name = name;
      this.feedback = feedback;


}

}
public class Main
```

```
{

    public static void main(String[] args)

{


        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        List<Customer> customers = new ArrayList<>();
        for (int i = 0; i < n; i++)
{


            String name = sc.next();
            int feedback = sc.nextInt();
            customers.add(new Customer(name, feedback));


}
        Consumer<Customer> display = c -> System.out.println("Customer: " +
c.name + ", Feedback Score: " + c.feedback);
        customers.forEach(display);
        double average = customers.stream().mapToInt(c ->
c.feedback).average().orElse(0);
        System.out.printf("Average Feedback: %.2f", average);


}

}
```

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