Rajalakshmi Engineering College

Name: Jathin Chowdary CH
Email: 240701209@rajalakshmi.edu.in
Roll no: 240701209
Phone: 8939767767
Branch: REC
Department: CSE - Section 10
Batch: 2028
Degree: B.E - CSE

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

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 scanner = new Scanner(System.in);

        double height = scanner.nextDouble();
        double weight = scanner.nextDouble();

        double bmi = weight / (height * height);

        String classification;
        if (bmi < 18.5) {
            classification = "Underweight";
        } else if (bmi >= 18.6 && bmi <= 24.9) {
            classification = "Normal Weight";
        } else if (bmi >= 25.0 && bmi <= 29.9) {
```

```
            classification = "Overweight";
        } else {
            classification = "Obese";
        }

        System.out.printf("BMI: %.2f\n", bmi);
        System.out.println("Classification: " + classification);

        scanner.close();
    }
}
```

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.Scanner;

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

      Scanner scanner = new Scanner(System.in);

      double a = scanner.nextDouble();
      double b = scanner.nextDouble();
      double c = scanner.nextDouble();
      scanner.close();

      double discriminant = (b * b) - (4 * a * c);

      if (discriminant > 0) {
         double sqrtD = Math.sqrt(discriminant);

         double root1 = (-b + sqrtD) / (2 * a);
         double root2 = (-b - sqrtD) / (2 * a);

         System.out.printf("Two real solutions:%nRoot1 = %.2f%nRoot2 = %.2f%n",
root1, root2);
      }
      else if (discriminant == 0) {
```

```
            double root = -b / (2 * a);
            System.out.printf("One real solution:%nRoot = %.2f%n", root);
        }
        else {
            System.out.println("There are no real solutions.");
        }
    }
}
```

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

3. Problem Statement

Ram wants to evaluate the time required to break even on an investment based on initial costs, monthly profits, and monthly expenses. Write a program to calculate the break-even point in months and categorize the return on investment.

Compute the break-even point by using the formula: initial cost / (monthly profit - monthly expenses)Based on the break-even point, classify the return on investment into one of the following categories:Quick Return: If the break-even point is 3 months or fewer.Average Return: If the break-even point is between 4 and 12 months, inclusive.Long-term Return: If the break-even point exceeds 12 months.

Ram is new to programming, so he seeks your assistance in creating the program.

Note: monthly profit is always greater than monthly expenses.

Input Format

The first line of input consists of a double value representing the initial cost.

The second line consists of a double value representing the monthly profit.

The third line consists of a double value representing the monthly expenses.

Output Format

The first line prints "Break-even Point:", followed by the break-even point as a decimal number (of double datatype), formatted to two decimal places.

The second line prints "Category: ",followed by the investment return as a String, which can be one of:

- "Quick Return" if break-even point ≤ 3
- "Average Return" if break-even point ≤ 12
- "Long-term Return" if break-even point > 12

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10000.50
5000.75
1000.10

Output: Break-even Point: 2.50
Category: Quick Return

Answer

```
import java.util.Scanner;

class BreakEvenPoint {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double initialCost = scanner.nextDouble();
        double monthlyProfit = scanner.nextDouble();
        double monthlyExpenses = scanner.nextDouble();

        double breakEvenPoint = initialCost / (monthlyProfit - monthlyExpenses);

        System.out.printf("Break-even Point: %.2f\n", breakEvenPoint);

        if (breakEvenPoint <= 3) {
            System.out.println("Category: Quick Return");
        } else if (breakEvenPoint <= 12) {
            System.out.println("Category: Average Return");
        } else {
            System.out.println("Category: Long-term Return");
        }
    }
```

}

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

4. 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.Scanner;

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

      int sumOfDigits = 0;
      int originalNumber = number;

      while (number > 0) {
         sumOfDigits += number % 10;
         number /= 10;
      }

      boolean isDivisible = originalNumber % sumOfDigits == 0;

      if (isDivisible) {
         System.out.println(originalNumber + " is divisible by the sum of its
digits.");
      } else {
         int closestSmallerNumber = -1;
         number = originalNumber - 1;

         while (number > 0) {
            if (number % sumOfDigits == 0) {
               closestSmallerNumber = number;
               break;
            }
```

```
            number--;
        }

        System.out.println(originalNumber + " is not divisible by the sum of its
digits.");
        if (closestSmallerNumber != -1) {
            System.out.println("The closest smaller number that is divisible: " +
closestSmallerNumber);
        }
    }
  }
}
```

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