Rajalakshmi Engineering College

Name: Gayathri Boopathy
Email: 240701141@rajalakshmi.edu.in
Roll no: 240701141
Phone: 9363837860
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

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

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

class main {
    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 netMonthly = monthlyProfit - monthlyExpenses;
        double breakEvenPoint = initialCost / netMonthly;
        DecimalFormat df = new DecimalFormat("0.00");
        String formattedBreakEven = df.format(breakEvenPoint);
        String category;
        if (breakEvenPoint <= 3) {
            category = "Quick Return";
        } else if (breakEvenPoint <= 12) {
            category = "Average Return";
        } else {
            category = "Long-term Return";
        }
        System.out.println("Break-even Point: " + formattedBreakEven);
        System.out.println("Category: " + category);
    }
}
```

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

2. 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

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

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

        int temp = N;
        int digitCount = 0;
        int digitSum = 0;

        do {
            int digit = temp % 10;
            digitSum += digit;
            digitCount++;
            temp /= 10;
        } while (temp > 0);

        if (digitCount == digitSum) {
            System.out.println("The number of digits in " + N + " matches the sum of
its digits.");
        } else {
            System.out.println("The number of digits in " + N + " does not match the
sum of its digits.");
        }
    }
}
```

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

3. 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

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

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 discriminant = b * b - 4 * a * c;

        if (discriminant > 0) {
            double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
            double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
            System.out.println("Two real solutions:");
            System.out.printf("Root1 = %.2f%n", root1);
            System.out.printf("Root2 = %.2f%n", root2);
        }
        else if (discriminant == 0) {
            double root = -b / (2 * a);
            System.out.println("One real solution:");
            System.out.printf("Root = %.2f%n", root);
        }
        else {
            System.out.println("There are no real solutions.");
        }
    }
}
```

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

4. 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

```
// You are using Java

import java.util.Scanner;

public 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 <= 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();
        }

        sc.close();
    }
}
```

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