

Rajalakshmi Engineering College

Name: giri prasad
Email: 240701147@rajalakshmi.edu.in
Roll no:
Phone: 9150391064
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

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 = $b^2 - 4ac$

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

2. 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

3. 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.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int number = scanner.nextInt();
        if (number % 5 == 0) {
            System.out.println(number + " is a multiple of 5");
        } else if (number % 7 == 0) {
            System.out.println(number + " is a multiple of 7");
        }
        else{
            System.out.println(number + " is neither multiple of 5 nor 7");
        }
        scanner.close();
    }
}
```

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.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