

# Rajalakshmi Engineering College

Name: Nikitha A  
Email: 241901073@rajalakshmi.edu.in  
Roll no: 241901073  
Phone: 7200177269  
Branch: REC  
Department: CSE (CS) - Section 2  
Batch: 2028  
Degree: B.E - CSE (CS)

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.*;
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();
        double dis = (b*b)-(4*a*c);
        if (dis > 0){
            double sqrtD = Math.sqrt(dis);
            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 (dis == 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.");
    }
    scanner.close();
}
}

```

**Status :** Correct

**Marks :** 10/10

## 2. 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:  $\text{initial cost} / (\text{monthly profit} - \text{monthly expenses})$ . 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  $\leq 3$
- "Average Return" if break-even point  $\leq 12$
- "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;
public class Main{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double Cost = scanner.nextDouble();
        double Profit = scanner.nextDouble();
        double Expenses = scanner.nextDouble();
        double Point = Cost / (Profit - Expenses);
        System.out.printf("Break-even Point: %.2f\n",Point);
        if (Point<= 3) {
            System.out.println("Category: Quick Return");
        } else if (Point <=12) {
            System.out.println("Category: Average Return");
        } else {
            System.out.println("Category: Long-term Return");
        }
        scanner.close();
    }
}
```

**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.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.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>0;i--) {
            for (int j=1; j<=i;j++) {
                System.out.print("* ");
            }
            System.out.println();
        }
        scanner.close();
    }
}
```

**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 scanner= new Scanner(System.in);
        int n =scanner.nextInt();
        int sum= 0;
        int num= n;
        while (n>0){
            sum +=n % 10;
            n /= 10;
        }
        boolean isDiv= num % sum == 0;
        if (isDiv){
            System.out.println(num + " is divisible by the sum of its digits.");
        }
        else{
            int closest= -1;
            n = num -1;
            while (n > 0){
                if (n% sum== 0) {
                    closest=n;
                    break;
                }
                n--;
            }
            System.out.println(num+" is not divisible by the sum of its digits.");
            if (closest !=-1) {
                System.out.println("The closest smaller number that is divisible: " +
closest);
            }
        }
        scanner.close();
    }
}
```



}

**Status :** Correct

**Marks : 10/10**