

# Rajalakshmi Engineering College

Name: Ganesh Raghav C  
Email: 240701138@rajalakshmi.edu.in  
Roll no:  
Phone: 6382988302  
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

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***

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

```
public class Main {
    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 % 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

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 =  $\text{weight}/(\text{height} \times \text{height})$

### ***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***

// You are using Java

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

```
public 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

### 3. Problem Statement

Noah is analyzing numbers within a given range [A, B] and wants to calculate a special sum. For each number in the range, he calculates the product of its odd digits (ignoring even digits). If the number contains no odd digits, it is skipped. The sum of these products for all numbers in the range is the result.

Write a program to compute this sum.

Example

Input:

10 12

Output:

3

Explanation:

For 10, odd digits = 1, product = 1.

For 11, odd digits = 1, 1, product =  $1 * 1 = 1$ .

For 12, odd digits = 1, product = 1.

Total sum =  $1 + 1 + 1 = 3$

#### ***Input Format***

The input consists of two space-separated integers A and B, representing the inclusive range boundaries.

#### ***Output Format***

The output prints a single integer representing the sum of the products of odd digits for all numbers in the range.

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: 10 12

Output: 3

### ***Answer***

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int A = sc.nextInt();
        int B = sc.nextInt();
        int totalSum = 0;

        for (int i = A; i <= B; i++) {
            int num = i;
            int product = 1;
            boolean hasOdd = false;

            while (num > 0) {
                int digit = num % 10;
                if (digit % 2 == 1) {
                    product *= digit;
                    hasOdd = true;
                }
                num /= 10;
            }

            if (hasOdd) {
                totalSum += product;
            }
        }

        System.out.println(totalSum);
    }
}
```

```
}  
}
```

**Status :** Correct

**Marks :** 10/10

#### 4. 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**

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

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

        double monthlyNetProfit = monthlyProfit - monthlyExpenses;
        double breakEvenPoint = initialCost / monthlyNetProfit;

        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**