Rajalakshmi Engineering College

Name: Prajith S
Email: 240701390@rajalakshmi.edu.in
Roll no:
Phone: 8610799150
Branch: REC
Department: CSE - Section 8
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;
public class Main{
   public static void main(String [] args){
      Scanner n=new Scanner(System.in);
      double i=n.nextDouble();
      double mp=n.nextDouble();
```

```
        double me=n.nextDouble();
        double be=i/(mp-me);
        System.out.printf("Break-even Point: %.2f\n",be);
        if(be<=3){
            System.out.println("Category : Quick Return");
        }
        else if(be>4 && be<12){
            System.out.println("Category : Average Return");
        }
        else if(be>=12){
            System.out.println("Category : Long-term Return");
        }


    }


}
```

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

2. 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

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

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

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

            while (temp > 0) {
                int digit = temp % 10;
```

```
            if (digit % 2 == 1) {
                prod *= digit;
                hasOdd = true;
            }
            temp /= 10;
        }

        if (hasOdd) totalSum += prod;
    }

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

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

3. 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.*;

class Main {
   public static void main(String[] args) {
      Scanner n = new Scanner(System.in);
      int n1 = n.nextInt();

      int sum = 0,np = n1;
      while (np > 0) {
         sum += np % 10;
         np /= 10;
      }

      if (n1 % sum == 0) {
         System.out.println(n1 + " is divisible by the sum of its digits.");
      } else {
```

```
            System.out.println(n1 + " is not divisible by the sum of its digits.");
            int c = n1- 1;
            while (c % sum != 0){
                c--;
            }
            System.out.println("The closest smaller number that is divisible: " + c);
        }
    }
}
```

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

4. 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

```
// You are using Java
import java.util.Scanner;
public class Main{
   public static void main(String[] args){
      Scanner n=new Scanner(System.in);
      int n1=n.nextInt();
      if(n1%7==0){
         System.out.println(n1+"is a multiple of 7");
      }
      else if(n1%5==0){
         System.out.println(n1+"is a multiple of 5");
      }
      else if(n1%7!=0 && n1%5!=0){
         System.out.println(n1+"is neither multiple of 5 nor 7");
      }
   }
}
```

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