Question 1) **Problem Statement**

Guna is developing a banking application that allows customers to deposit money into their accounts. The bank provides each customer with an initial balance of **50** rupees. To facilitate the deposit process, Guna has created a class called **AddAmount**.

This class represents a transaction to add a specific amount of money to a customer's account. After the transaction is completed, the total amount in the customer's account is printed.

**Input format :**

The input consists of a single integer, representing the additional amount to be deposited into the customer's account (the deposit amount).

**Output format :**

The output displays the total amount in the customer's account after the deposit.

**Refer to the sample output for formatting specifications.**

**Code constraints :**

The input deposit amount is a non-negative integer.

**Sample test cases :**

**Input 1 :**

400

**Output 1 :**

450

**Input 2 :**

250

**Output 2 :**

300

Question 2) **Single File Programming Question**

**Problem statement**

**﻿**

Design a banking system to handle account transactions. The system should allow users to deposit and withdraw money from their accounts. Each account is associated with an account holder's name, account number, and balance.

Implement the system using a union named **AccountData** that contains a structure with fields for the account holder's name, account number, and balance.

**Input format :**

The first line consists of the account holder's name as a string.

The second line consists of the account number as an integer.

The third line consists of the initial balance as a double value.

The fourth line consists of the amount to deposit as a double value.

The fifth line consists of the amount to withdraw as a double value.

**Output format :**

The first line of output prints "Amount deposited successfully." after processing the deposit.

The second line of output prints:

1. "Amount withdrawn successfully." if the withdrawal is successful.
2. "Insufficient balance. Withdrawal failed." if the balance is insufficient.

The third line of output prints "Account details:".

The fourth line of output prints "Account holder's name: <name>", where <name> is the account holder's name.

The fifth line of output prints "Account number: <number>", where <number> is the account number.

The sixth line of output prints "Balance: <balance>", where <balance> is the remaining balance after transactions, formatted to two decimal places.

**Refer to sample input and output specifications**

**Code constraints :**

The account holder's name should not exceed 50 characters.

The initial balance, deposit amount, and withdrawal amount should be non-negative double values.

**Sample test cases :**

**Input 1 :**

John Doe

123456

5000

1000

500

**Output 1 :**

Amount deposited successfully.

Amount withdrawn successfully.

Account details:

Account holder's name: John Doe

Account number: 123456

Balance: 5500.00

**Input 2 :**

Alice Smith

987654

8000

1500

10000

**Output 2 :**

Amount deposited successfully.

Insufficient balance. Withdrawal failed.

Account details:

Account holder's name: Alice Smith

Account number: 987654

Balance: 9500.00

Question 3) **Problem Statement**

You are developing a module for a text-processing application that will manipulate integers. The module should provide functionalities to set an integer, count the number of digits, find the number of odd digits, and calculate the sum of odd digits in the integer.

You are tasked with implementing an **IntegerManipulator** class that performs various operations on a given integer. The class should provide methods to count the number of digits, find the number of odd digits, and calculate the sum of odd digits in the integer.

Use a Non-inline member function.

**Note:** This is a sample question asked in a mPhasis interview.

**Input format :**

The input consists of an integer n.

**Output format :**

The first line displays the number of digits in the number.

The second line displays the number of odd digits in the integer.

**Refer to the sample output for formatting specifications.**

**Code constraints :**

1 ≤ n ≤ 109

**Sample test cases :**

**Input 1 :**

1124

**Output 1 :**

Number of digits: 4

Number of odd digits: 2

**Input 2 :**

2468

**Output 2 :**

Number of digits: 4

Number of odd digits: 0

Question 4)

**Problem Statement**

Shabu works at a library and needs to analyze the frequency of digits in identification numbers. Help Shabu by writing a program that counts how many times each digit (0-9) appears in a given number. The output should use the **setw** manipulator function for formatting.

**Note:** Make the width length as 2.

**Input format :**

The input consists of an integer N, representing the identification number.

**Output format :**

For each digit (0-9) that appears in the number, output the digit and the number of times it appears, formatted with a width of 2 for the count.

The output should be in the format: "Digit X: Y times", where X is the digit and Y is its count.

**Refer to the sample outputs for the formatting specifications.**

**Code constraints :**

102≤ N ≤ 109

**Sample test cases :**

**Input 1 :**

12312

**Output 1 :**

Digit 1: 2 times

Digit 2: 2 times

Digit 3: 1 times

**Input 2 :**

123

**Output 2 :**

Digit 1: 1 times

Digit 2: 1 times

Digit 3: 1 times

**Input 3 :**

900

**Output 3 :**

Digit 0: 2 times

Digit 9: 1 times

Question 5) **Problem Statement**

Alex is a math enthusiast who loves playing with numbers. One day, Alex came across a challenge to write a program that reverses a given number. Intrigued by the challenge, Alex decides to write a program using reference variables.

Can you help Alex by creating a program that reverses the number using reference variables?

**Function Name:** void reverseNumber

**Note:** This is a sample question asked in a HCL interview.

**Input format :**

The input consists of a positive integer N.

**Output format :**

The output displays the reversed number of N.

**Code constraints :**

1 ≤ N ≤ 107

**Sample test cases :**

**Input 1 :**

157

**Output 1 :**

751

**Input 2 :**

120

**Output 2 :**

21