**Structured programming basics**

**Section A**

**Questions and their solutions:**

1. **Write a C program to SWAP (interchange) 2 numbers without using third variable (filename: two\_swap.c)**

**README:**

Swapping Values Without Using a Third Variable

**Description:**

This C program demonstrates how to swap the values of two variables without using a third variable. The program takes two integer values, a and b, and swaps their values using arithmetic operations.

* Initialize two integer variables a and b
* To swap the values:
* Add **a** and **b** and store the result in **a *(a = a + b).***
* Subtract the original value of b from the updated value of a and store the result in ***b (b = a - b).***
* Subtract the original value of a from the updated value of b and store the result in **a *(a = a - b).***
* Print the swapped values of a and b.

**Compilation and Execution:**

* Save the code to a file named **swap\_values.c.**
* Compile the code using the following command: gcc two\_swap.c -o swap
* Execute the compiled binary: ./swap
* The output will display the swapped values of **a** and **b**.

1. **Write a C program to calculate the remainder of 2 numbers without using % operator (filename: remainder.c)**

**README:**

To Calculate Remainder Without Using % Operator

**Description:**

This C program demonstrates how to calculate the remainder of two integers without using the modulo operator (%). It defines a function calculate remainder that iteratively subtracts the divisor from the dividend until the dividend becomes less than the divisor. The remaining value is the remainder.

Code Explanation

The program includes the standard input/output library ***(<stdio.h>).***

The calculate\_remainder function takes two integer arguments ***(a and b)*** and returns the remainder of **a** divided by **b**. Inside the calculate\_remainder function:

A while loop is used to repeatedly subtract b from a until a becomes less than b.

The final value of **a** after the loop is the remainder.

In the main function:

The user is prompted to enter two numbers ***(a and b).***

The calculate\_remainder function is called with the input values.

The remainder is printed to the console.

Usage

Compilation and Execution

Save the code to a file named **remainder.c**

gcc remainder.c -o rem

Run the compiled executable

(./rem).

Enter two integer values when prompted.

The program will calculate and display the remainder.

Example

Suppose you compile and run the program, and you input a = 15 and b = 4. The output will be:

Enter two numbers: 15 4

Remainder: 3

1. **Write a C program to calculate the sum of digits of a three-digit number e.g.125 is 8 (filename: sum.c)**

**README:**

**Description:**

This C program calculates the sum of digits of a three-digit number entered by the user. It prompts the user to input a three-digit number, extracts the individual digits, and computes their sum.

**Explanation:**

* The program starts by declaring necessary variables: number, digit1, digit2, digit3, and sum\_of\_digits.
* The user is prompted to input a three-digit number using printf.
* The input is read using scanf and stored in the number variable.
* The program extracts the individual digits using the following formulas:
* digit1 = number / 100 (to get the hundreds place digit)
* digit2 = (number / 10) % 10 (to get the tens place digit)
* digit3 = number % 10 (to get the ones place digit)
* The sum of digits is calculated as sum\_of\_digits = digit1 + digit2 + digit3.
* Finally, the program displays the result using printf.

**Example:** *Suppose the user enters the number “456”:*

* digit1 will be “4” (hundreds place digit)
* digit2 will be “5” (tens place digit)
* digit3 will be “6” (ones place digit)
* The sum of digits will be “15”.

**Compilation and Execution:**

* Save the code to a file named **sum.c**
* Compile the code using the following command: gcc sum.c -o sum
* Execute the compiled binary: ./sum

1. **Write a C program to print the reverse of a 3-digit number. (filename: reverse.c)**

**README:**

Reverse of a 3-Digit Number

This C program reads a 3-digit number from the user and prints its reverse.

* How to Compile and Run
* Make sure you have a C compiler installed (e.g., gcc).
* Open a terminal or command prompt.
* Navigate to the directory containing the reverse.c file.
* Compile the program using the following command:

bash

gcc reverse -o rev reverse.c

* Run the compiled program:

**Program Description**

* The program prompts the user to enter a 3-digit number.
* It reads the input from the user.
* The program calculates the reverse of the number.
* Finally, it prints the reversed number.

**For Example**

Enter a 3-digit number: 123

Reversed number: 321

**Section B**

**Problem 2:**

**README:**

Credit Card Balance Calculator

This program calculates the minimum fixed monthly payment needed to pay off a credit card balance within **12 months.**

**Variables:**

* balance: The outstanding balance on the credit card.
* annualInterestRate: The annual interest rate as a decimal.

**Calculation:**

The program calculates the monthly interest rate, monthly unpaid balance, and updated balance each month using the following formulas:

* Monthly interest rate = (Annual interest rate) / 12.0
* Monthly unpaid balance = (Previous balance) - (Minimum fixed monthly payment)
* Updated balance each month = (Monthly unpaid balance) + (Monthly interest rate x Monthly unpaid balance)

Compilation and Execution

* Save the code to a file named **problem\_2.c**
* Compile the code using the following command:

gcc problem\_2.c -o problem

* Execute the compiled binary: ./problem

* The output will display the **lowest payment**.

**Output:**

The program prints out the lowest monthly payment that will pay off all debt in under 1 year. The monthly payment is a multiple of 10FCFA and is the same for all months. It is possible for the balance to become negative using this payment scheme, which is okay.

**Test Cases:**

Test Case 1:

* balance = 3329
* annualInterestRate = 0.2

**Result Your Code Should Generate: Lowest Payment**: 310

**Test Case 2:**

* balance = 4773
* annualInterestRate = 0.2

**Result Your Code Should Generate:** Lowest Payment: 440

**Test Case 3:**

* balance = 3926
* annualInterestRate = 0.2

**Result Your Code Should Generate:** Lowest Payment: 360