Practice Problem

Section 1: Print & Read

This section covers the fundamentals of input/output in C, variable operations, data types, assignments, and basic arithmetic.

Basic Input/Output and Variable Handling

1. Print "**Hello world…**"
2. Print an integer constant.
3. Read an integer from the user and print it.
4. Read two integers from the user and display both.
5. Read two integers, to perform :
    a. addition, subtraction, and multiplication **without using a third variable**
    b. addition, subtraction, and multiplication using a third variable
6. Swap two numbers using a third variable.
7. Swap two numbers without using a third variable.
8. Get and display the size of `int`, `float`, `double`, and `char` using `sizeof`.
9. Assign a character constant to a variable and print it.
10. Read a character from the user, assign to a variable, and print it.
11. Read the string "Hello World" using `scanf` and print it. (Explore why this may fail.)

Arithmetic and Expression Evaluation

12. Accept an integer x and compute :
    a. x^2
    b. $x^2 + 2x$
13. Accept two integer $x,\ y$ and compute :
    a. $x^3 + 3x^2 + 4x - y^3$
    b. $\sqrt{2x^2 + 4y^2 + x^3 + 10}$
    c. $\frac{\sqrt{4x^2 + 8y^2 + x^3 + 5}}{2x^2}$
14. Find the roots of a quadratic equation given coefficients a, b and c.
15. Check what happens if non-numeric data is entered when an integer or float is expected (input validation).

Section 2: Conditional Statements

This section provides practice programs focusing on conditional statements, thoroughly covering `if`, `if...else`, `else if` ladders, `switch`, and the conditional (ternary) operator

2.1 Simple If, If...Else, and Else If

1. Read an integer and determine whether it is positive, negative, or zero.
2. Read two integers and check if they are equal, or determine which is larger.
3. Read three integers and find the largest among them.
4. Check if a number is odd or even.
5. Determine if a given year is a leap year.
6. Check if a character entered by the user is an alphabet or not.
7. Check if a character is a vowel or consonant.
8. Check if a character is uppercase or lowercase.
9. Check if the input character is the alphabet, digit, or special character.

2.2 Nested If...Else and Logical Operators

10. Read marks for five subjects and assign a grade based on percentage ranges (A/B/C/D/E/F).
11. Compute profit or loss given cost price and selling price; display the result.
12. Read three angles of a triangle and check whether the triangle is valid:
a. Each angle must be greater than 0°.
b. The sum of all three angles must be exactly 180°.
13. Write a program that reads three positive numbers a, b and c which represent the lengths of the sides of a triangle. Check if they form a valid triangle and categorize it (equilateral, isosceles, scalene).
    a. *a+b > c and a+c > b and b+c > a implies a valid triangle*
    b. *a=b=c => equilateral, a=b or a=c or b=c => isosceles, a!=b!=c => scalene*
14. Input week number (1–7) and print corresponding weekday name.
15. Input month number and display the number of days in the month (consider leap years for February).

2.3 Ladder and Multiple Conditions

16. Print message based on multiple ranges:
    a. If number <100 print “small”,
    b. 100–200 “large”,
    c. 201–300 “bigger”,
    d. 301–400 “largest”,
    e. >400 “very large”

2.4 Conditional (Ternary) Operator

17. Input two numbers, find the maximum using the conditional operator.
18. Use a ternary operator to find if a number is even or odd.
19. Assign remarks ("Pass"/"Fail") using conditional operators based on marks.
20. Use nested conditional operators to classify a number as positive/negative/zero.

2.5 Switch Statements

21. Read a character (`+`, `-`, `*`, `/`) and two integers, and perform the corresponding arithmetic operations.
22. Take a number (1–7) and print the day of the week using a switch.
23. Input a number (1–12) and display the name of the month via switch.
24. Read a grade character and print remarks based on the grade using switch-case.
25. Given a number, output "small" if less than 10, "large" if more than 10, "equal" if exactly 10.

Section 3: Loops and Iteration

This section develops your mastery of loops and iterative constructs

3.1 While Loops

1. Print all numbers from 1 to n (user input)
2. Print all even numbers between 1 and n (user input)
3. Print the sum of all integers from 1 to n (user input)
4. Print the sum of all odd numbers from 1 to n (user input)
5. Calculate and print the sum of the series 1 + 1/2 + 1/3 + ... + 1/n
6. Sum of squares of first n numbers
7. Print the multiplication table for a given number
8. Reverse a number entered by the user
9. Count and display the number of digits in a number
10. Calculate the **factorial** of a number
11. Print the first n terms of the **Fibonacci** series
12. Check whether a given number is prime or composite
13. Find the power (x^y) without using *pow()*
14. Display all uppercase ASCII characters with their integer values
15. Print a square/triangle/star pattern based on user input (height)
16. Check if a number is a **palindrome**
17. Find the **GCD (HCF) and LCM** of two numbers
18. Implement all above programs using **for** loop

3.2 Do-While Loops

1. Read numbers until the user enters 0; compute and print their sum using a do-while loop.
2. Ask the user repeatedly for a password until correct (demonstrating do-while).
3. Print a menu-driven calculator: repeat operations until the user chooses to exit (do-while).
4. Display digits of a number, one per line, using a do-while loop.

3.3 Nested Loops

1. Print multiplication tables from 2 to 11 using nested for loops.
2. Find and display all prime numbers between 1 and 1000000
3. Generate and print the prime-factored form of numbers 1 to n.
4. Check and print all Armstrong numbers in a range using nested loops.
5. Write a program to determine whether two given numbers are amicable numbers
    a. Two numbers are said to be **amicable** if the **sum of proper divisors** (excluding the number itself) of each number equals the **other number** - (220/284)
6. Write a program to check whether a given number is a perfect number
    a. A **perfect number** is a positive integer that is equal to the **sum of its proper divisors** (excluding itself) e.g : 28: 1, 2, 4, 7, 14 → sum = 28
7. Print various formatted patterns (pyramid, diamond, Pascal's triangle) using nested loops.

3.4 Use of Break and Continue

1. Skip printing odd numbers in a loop using continue.
2. Program to keep summing until the user enters a negative number, then break the loop and print the sum.
3. Find the smallest divisor of a number greater than 1 using break.
4. Find the first number greater than n that is divisible by 7.
5. Print numbers from 1 to N, skipping numbers that are divisible by 3.
6. Allow the user to enter a password up to 3 times. If the correct password is entered, print Access granted and break. Otherwise, after 3 attempts, print Account locked.
7. Print numbers from 1 to N, but skip those that end in 5 using continue.
8. Print the smallest number less than or equal to N that is divisible by both 4 and 6.

3.6 Loops with Complex Conditions