C Programming Problems Using Loops

For Loop Problems (40 Problems: Easy to Hard)

Easy (1–10)

1. **Sum of First N Numbers**: Calculate the sum of the first N natural numbers using a `for` loop.
   Sample Input:

   ```
   5
   ```

   Sample Output:

   ```
   Sum: 15
   ```

2. **Print Numbers 1 to N**: Print all numbers from 1 to N using a `for` loop.
   Sample Input:

   ```
   5
   ```

   Sample Output:

   ```
   1 2 3 4 5
   ```

3. **Print Even Numbers**: Display all even numbers from 1 to N using a `for` loop.
   Sample Input:

   ```
   10
   ```

   Sample Output:

   ```
   2 4 6 8 10
   ```

4. **Print Odd Numbers**: Display all odd numbers from 1 to N using a `for` loop.
   Sample Input:

   ```
   10
   ```

   Sample Output:

   ```
   1 3 5 7 9
   ```

5. **Multiplication Table**: Print the multiplication table of a given number up to 10 using a `for` loop.
   Sample Input:

   ```
   3
   ```

   Sample Output:

   ```
   3 x 1 = 3
   3 x 2 = 6
   ...
   3 x 10 = 30
   ```

6. **Sum of Even Numbers**: Calculate the sum of even numbers from 1 to N using a `for` loop.
   Sample Input:

   ```
   10
   ```

   Sample Output:

   ```
   Sum of even numbers: 30
   ```

7. **Sum of Odd Numbers**: Calculate the sum of odd numbers from 1 to N using a `for` loop.
   Sample Input:

   ```
   10
   ```

Sample Output:

```
Sum of odd numbers: 25
```

8. **Count Multiples of 3**: Count multiples of 3 from 1 to N using a `for` loop.
   Sample Input:

   ```
   15
   ```

   Sample Output:

   ```
   Multiples of 3: 5
   ```

9. **Print Squares**: Print the squares of numbers from 1 to N using a `for` loop.
   Sample Input:

   ```
   5
   ```

   Sample Output:

   ```
   1 4 9 16 25
   ```

10. **Factorial Calculation**: Compute the factorial of a given number N using a `for` loop.
    Sample Input:

    ```
    5
    ```

    Sample Output:

    ```
    Factorial of 5: 120
    ```

Medium (11–25)

11. **Sum of Squares**: Calculate the sum of squares of the first N natural numbers using a `for` loop.
    Sample Input:

    ```
    3
    ```

Sample Output:

```
Sum of squares: 14
```

12. **Sum of Cubes**: Calculate the sum of cubes of the first N natural numbers using a `for` loop.
    Sample Input:

    ```
    3
    ```

    Sample Output:

    ```
    Sum of cubes: 36
    ```

13. **Fibonacci Sequence**: Generate the first N terms of the Fibonacci sequence using a `for` loop.
    Sample Input:

    ```
    6
    ```

    Sample Output:

    ```
    Fibonacci sequence: 0 1 1 2 3 5
    ```

14. **Count Digits**: Count the number of digits in a given integer using a `for` loop.
    Sample Input:

    ```
    54321
    ```

    Sample Output:

    ```
    Number of digits: 5
    ```

15. **Sum of Digits**: Calculate the sum of digits of a given number using a `for` loop.
    Sample Input:

    ```
    123
    ```

Sample Output:

```
Sum of digits: 6
```

16. **Reverse a Number**: Reverse the digits of a given number using a `for` loop.
    Sample Input:

    ```
    1234
    ```

    Sample Output:

    ```
    Reversed number: 4321
    ```

17. **Power Calculation**: Compute base^{exponent} for user-input base and exponent using a `for` loop.
    Sample Input:

    ```
    2 3
    ```

    Sample Output:

    ```
    2^3 = 8
    ```

18. **Star Pyramid**: Print a centered pyramid of stars (row i has 2i-1 stars) with N rows using nested `for` loops.
    Sample Input:

    ```
    4
    ```

    Sample Output:

    ```
       *
      ***
     *****
    *******
    ```

19. **Number Pyramid**: Print a pyramid where each row contains the row number repeated using nested `for` loops.
    Sample Input:

```
3
```

Sample Output:

```
  1
 2 2
3 3 3
```

20. **Inverted Star Pyramid**: Print an inverted pyramid of stars (decreasing star count) with N rows using nested `for` loops.
    Sample Input:

    ```
    3
    ```

    Sample Output:

    ```
    *****
     ***
      *
    ```

21. **Half Pyramid of Numbers**: Print a half pyramid of numbers (e.g., row 1: 1, row 2: 1 2) using nested `for` loops.
    Sample Input:

    ```
    4
    ```

    Sample Output:

    ```
    1
    1 2
    1 2 3
    1 2 3 4
    ```

22. **Triangular Number Sequence**: Generate the first N triangular numbers (e.g., 1, 3, 6, 10) using a `for` loop.
    Sample Input:

    ```
    4
    ```

    Sample Output:

```
Triangular numbers: 1 3 6 10
```

23. **Sum of Multiples of 3 or 5**: Sum all multiples of 3 or 5 below N using a `for` loop.
    Sample Input:

    ```
    10
    ```

    Sample Output:

    ```
    Sum of multiples: 23
    ```

24. **Pascal's Triangle**: Print Pascal's triangle for N rows, calculating each element using a `for` loop.
    Sample Input:

    ```
    4
    ```

    Sample Output:

    ```
       1
      1 1
     1 2 1
    1 3 3 1
    ```

25. **Diamond Star Pattern**: Print a diamond pattern of stars (increasing then decreasing rows) using nested `for` loops.
    Sample Input:

    ```
    3
    ```

    Sample Output:

    ```
      *
     ***
    *****
     ***
      *
    ```

Hard (26–40)

26. **Square Number Pattern**: Print a square pattern of sequential numbers (e.g., 3x3 grid: 1 to 9) using a `for` loop.
    Sample Input:

    ```
    3
    ```

    Sample Output:

    ```
    1 2 3
    4 5 6
    7 8 9
    ```

27. **Butterfly Pattern with Stars**: Print a butterfly pattern using stars with nested `for` loops.
    Sample Input:

    ```
    3
    ```

    Sample Output:

    ```
    *     *
    **   **
    *** ***
    **   **
    *     *
    ```

28. **Rhombus Pattern**: Print a rhombus pattern of stars using nested `for` loops.
    Sample Input:

    ```
    4
    ```

    Sample Output:

    ```
       ****
      ****
     ****
    ****
    ```

29. **Hollow Star Pyramid**: Print a hollow pyramid of stars (only border stars) with N rows using nested `for` loops.

Sample Input:

```
4
```

Sample Output:

```
   *
  * *
 *   *
*******
```

30. **Full Pyramid of Numbers**: Print a full centered pyramid of numbers using nested `for` loops.
    Sample Input:

    ```
    3
    ```

    Sample Output:

    ```
      1
     1 2 1
    1 2 3 2 1
    ```

31. **Arithmetic Sequence Sum**: Calculate the sum of an arithmetic sequence with first term a, common difference d, and N terms using a `for` loop.
    Sample Input:

    ```
    2 3 4
    ```

    Sample Output:

    ```
    Sum: 20
    ```

32. **Geometric Sequence Sum**: Calculate the sum of a geometric sequence with first term a, ratio r, and N terms using a `for` loop.
    Sample Input:

    ```
    2 2 4
    ```

    Sample Output:

```
Sum: 30
```

33. **Print Factorials Up to N**: Print factorials of numbers from 1 to N using a `for` loop.
Sample Input:

```
4
```

Sample Output:

```
Factorials: 1 2 6 24
```

34. **Hollow Diamond Pattern**: Print a hollow diamond pattern of stars using nested `for` loops.
Sample Input:

```
3
```

Sample Output:

```
  *
 * *
*   *
 * *
  *
```

35. **Number Triangle Pattern**: Print a triangle pattern where each row contains numbers from 1 to row number, reversed, using nested `for` loops.
Sample Input:

```
4
```

Sample Output:

```
   1
  2 1
 3 2 1
4 3 2 1
```

36. **Centered Number Square**: Print a square pattern where each row contains centered numbers (e.g., row i has i as the center number) using nested

`for` loops.
Sample Input:

```
3
```

Sample Output:

```
3 2 3
2 1 2
3 2 3
```

37. **Sum of Factorials**: Calculate the sum of factorials from 1! to N! using a `for` loop.
    Sample Input:

    ```
    3
    ```

    Sample Output:

    ```
    Sum of factorials: 9
    ```

38. **Spiral Number Pattern**: Print a spiral number pattern (e.g., row 1: ”1”, row 2: ”2 3 2”) using nested `for` loops.
    Sample Input:

    ```
    3
    ```

    Sample Output:

    ```
      1
     2 3 2
    3 4 5 4 3
    ```

39. **Print Multiples of K**: Print the first N multiples of a given number K using a `for` loop.
    Sample Input:

    ```
    3 5
    ```

    Sample Output:

```
Multiples: 3 6 9 12 15
```

40. **Checkerboard Star Pattern**: Print a checkerboard pattern of stars (alternating stars and spaces) using nested `for` loops.
    Sample Input:

    ```
    4
    ```

    Sample Output:

    ```
    * * *
     * * *
    * * *
     * * *
    ```

While Loop Problems (20 Problems)

1. **Sum Until Threshold**: Sum numbers entered by the user until the sum exceeds 100 using a `while` loop.
   Sample Input:

   ```
   50
   30
   25
   ```

   Sample Output:

   ```
   Sum: 105
   ```

2. **Number Guessing Game**: Implement a game where the user guesses a number (1–100) until correct, using a `while` loop.
   Sample Input: (Assume target is 42)

   ```
   50
   40
   42
   ```

   Sample Output:

   ```
   Too high!
   ```

```
Too low!
Correct! Number is 42
```

3. **Palindrome Number Check**: Check if a number is a palindrome (e.g., 121) using a `while` loop.
   Sample Input:

   ```
   121
   ```

   Sample Output:

   ```
   121 is a palindrome
   ```

4. **GCD Calculation**: Find the Greatest Common Divisor of two numbers using a `while` loop (Euclidean algorithm).
   Sample Input:

   ```
   48 18
   ```

   Sample Output:

   ```
   GCD: 6
   ```

5. **Binary Conversion**: Convert a decimal number to its binary equivalent using a `while` loop.
   Sample Input:

   ```
   13
   ```

   Sample Output:

   ```
   Binary: 1101
   ```

6. **Digit Extraction**: Extract and print each digit of a number in reverse order using a `while` loop.
   Sample Input:

   ```
   456
   ```

Sample Output:

```
Digits: 6 5 4
```

7. **Prime Number Check**: Check if a given number is prime using a `while` loop to test divisibility.
   Sample Input:

   ```
   17
   ```

   Sample Output:

   ```
   17 is prime
   ```

8. **Armstrong Number Check**: Check if a number is an Armstrong number (e.g., $153 = 1^3 + 5^3 + 3^3$) using a `while` loop.
   Sample Input:

   ```
   153
   ```

   Sample Output:

   ```
   153 is an Armstrong number
   ```

9. **Sum of Even Digits**: Sum the even digits of a number using a `while` loop.
   Sample Input:

   ```
   246
   ```

   Sample Output:

   ```
   Sum of even digits: 12
   ```

10. **Perfect Number Check**: Check if a number is perfect (sum of proper divisors equals the number) using a `while` loop.
    Sample Input:

    ```
    28
    ```

Sample Output:

```
28 is a perfect number
```

11. **Prime Factorization**: Find and print all prime factors of a given number using a `while` loop.
    Sample Input:

    ```
    100
    ```

    Sample Output:

    ```
    Prime factors: 2 2 5 5
    ```

12. **Happy Number Check**: Check if a number is a happy number (repeated sum of squares of digits leads to 1) using a `while` loop.
    Sample Input:

    ```
    19
    ```

    Sample Output:

    ```
    19 is a happy number
    ```

13. **Count Primes Up to N**: Count the number of prime numbers up to N using a `while` loop.
    Sample Input:

    ```
    20
    ```

    Sample Output:

    ```
    Number of primes: 8
    ```

14. **Sum of Primes Up to N**: Calculate the sum of all prime numbers up to N using a `while` loop.
    Sample Input:

    ```
    10
    ```

Sample Output:

```
Sum of primes: 17
```

15. **Triangular Number Check**: Check if a number is triangular (sum of first k natural numbers) using a `while` loop.
    Sample Input:

    ```
    6
    ```

    Sample Output:

    ```
    6 is a triangular number
    ```

16. **Collatz Sequence Length**: Find the length of the Collatz sequence for a given number (divide by 2 if even, multiply by 3 and add 1 if odd, until 1) using a `while` loop.
    Sample Input:

    ```
    6
    ```

    Sample Output:

    ```
    Sequence length: 8
    ```

17. **Power of Two Check**: Check if a number is a power of 2 using a `while` loop.
    Sample Input:

    ```
    16
    ```

    Sample Output:

    ```
    16 is a power of 2
    ```

18. **Sum Until Negative**: Sum numbers entered by the user until a negative number is entered using a `while` loop.
    Sample Input:

    ```
    5
    ```

```
10
-1
```

Sample Output:

```
Sum: 15
```

19. **Find First N Primes**: Print the first N prime numbers using a `while` loop.
    Sample Input:

    ```
    4
    ```

    Sample Output:

    ```
    Primes: 2 3 5 7
    ```

20. **Sum Until Even**: Sum numbers entered by the user until an even number is entered using a `while` loop.
    Sample Input:

    ```
    3
    5
    4
    ```

    Sample Output:

    ```
    Sum: 12
    ```

Do-While Loop Problems (5 Problems)

1. **Positive Number Validation**: Prompt the user to enter a positive number (1–100) and keep asking until a valid input is received, using a `do-while` loop.
   Sample Input:

   ```
   -5
   150
   42
   ```

Sample Output:

```
Enter a number (1-100): Invalid input! Try again.
Enter a number (1-100): Invalid input! Try again.
Enter a number (1-100): Valid number: 42
```

2. **Sum of User Inputs Until Zero**: Sum numbers entered by the user until they enter 0, ensuring at least one input is processed, using a `do-while` loop.
   Sample Input:

   ```
   10
   20
   0
   ```

   Sample Output:

   ```
   Enter a number (0 to stop): Enter a number (0 to stop):
   Enter a number (0 to stop): Sum: 30
   ```

3. **Menu-Driven Calculator**: Implement a calculator for basic operations (add, subtract, multiply, divide) based on user choice, continuing until the user chooses to exit, using a `do-while` loop.
   Sample Input:

   ```
   1
   5 3
   2
   5 3
   0
   ```

   Sample Output:

   ```
   Menu: 1=Add, 2=Subtract, 3=Multiply, 4=Divide, 0=Exit
   Enter choice: Enter two numbers: Result: 8
   Menu: 1=Add, 2=Subtract, 3=Multiply, 4=Divide, 0=Exit
   Enter choice: Enter two numbers: Result: 2
   Menu: 1=Add, 2=Subtract, 3=Multiply, 4=Divide, 0=Exit
   Enter choice: Exiting...
   ```

4. **Range Input Validation**: Prompt for a number in a specific range (10–50) and re-prompt for invalid inputs, using a `do-while` loop.

Sample Input:

```
5
60
42
```

Sample Output:

```
Enter a number (10-50): Invalid input! Try again.
Enter a number (10-50): Invalid input! Try again.
Enter a number (10-50): Valid number: 42
```

5. **Menu for Number Properties**: Provide a menu to compute the square, cube, or double of a number, repeating until the user exits, using a `do-while` loop.
Sample Input:

```
1
4
2
3
0
```

Sample Output:

```
Menu: 1=Square, 2=Cube, 3=Double, 0=Exit
Enter choice: Enter number: Square: 16
Menu: 1=Square, 2=Cube, 3=Double, 0=Exit
Enter choice: Enter number: Cube: 27
Menu: 1=Square, 2=Cube, 3=Double, 0=Exit
Enter choice: Exiting...
```