C++ PRACTICAL ASSIGNMENT

Level: Medium
Programming Language: C++
Focus Areas: Logical Thinking, Core Programming, Functions, and Loops
Restrictions:
- Do **not** use STL (vector, map, set, etc.)
- Do **not** use advanced or external libraries
- Use only basic C++ concepts (variables, loops, functions, arrays, classes)

Special Number Definitions (For Reference)

- **Palindrome Number:** A number that reads the same forward and backward (e.g., 121, 3443).
- **Armstrong Number:** A number equal to the sum of its digits raised to the power of the number of digits (e.g., 153).
- **Harshad Number:** A number divisible by the sum of its digits (e.g., 18 → 1+8=9, 18 is divisible by 9).
- **Spy Number:** A number whose sum of digits equals the product of its digits (e.g., 123 → sum=6, product=6).
- **Automorphic Number:** A number whose square ends with the same digits as the number itself (e.g., 25 → 625).
- **Duck Number:** A number that contains at least one zero but does not start with zero (e.g., 102).
- **Neon Number:** A number where the sum of digits of its square equals the original number (e.g., 9 → 81 → 8+1=9).
- **Strong Number:** A number equal to the sum of the factorials of its digits (e.g., 145).

SECTION A: FUNCTION-BASED PROBLEM SOLVING

(Each problem must be solved using a separate user-defined function)

1. Write a function to determine whether a given number is a **Palindrome Number**.
2. Write a function that calculates the **repeated sum of digits** of a number until it becomes a single digit.
3. Write a function to verify whether a number is an **Armstrong Number**.
4. Write a function to find the **largest digit** present in a given number.
5. Write a function to calculate **x raised to the power n (x^n)** without using the built-in `pow()` function.
6. Write a function to **count even digits and odd digits** in a given number.
7. Write a function to check whether a number is a **Harshad Number**.
8. Write a function that **reverses a number** and returns the reversed value.
9. Write a function to calculate the **product of digits** of a number.
10. Write a function to check whether a number is a **Spy Number**.
11. Write a function to determine whether a number is an **Automorphic Number**.

12. Write a function to **count the total number of digits** in a given number.
13. Write a function to check whether a number is a **Duck Number**.
14. Write a function to find the **difference between the sum of even digits and odd digits**.
15. Write a function to check whether a number is a **Neon Number**.

SECTION B: LOOP-BASED LOGICAL CHALLENGES

(All problems must be solved using loops only)

1. Print all **Palindrome Numbers** between 1 and N.
2. Display all **Armstrong Numbers** between two given numbers.
3. Count the number of digits in a number **without using any built-in function**.
4. Reverse a number using loops and **compare it with the original number**.
5. Print all numbers whose **sum of digits is a prime number**.
6. Print all **Strong Numbers** between 1 and N.
7. Find the **sum of digits present at even positions** (from right to left).
8. Print all numbers that contain **only odd digits**.
9. Count how many numbers between 1 and N have an **even sum of digits**.
10. Find the **LCM of two numbers** using loops.
11. Print numbers in which the **first and last digits are the same**.
12. Find the **repeated sum of digits** until the number becomes a single digit.
13. Print all numbers whose **reverse is divisible by 3**.
14. Print numbers that become **Palindrome after one reverse-and-add operation**.
15. Print a **numeric pyramid pattern** using loops.

Instructions for Students

- Follow proper coding standards and indentation
- Write comments for better readability
- Avoid copying code; understand the logic before implementation
- Each question should be written and tested independently

End of Assignment