1. **Scenario:** You are developing a banking application that categorizes transactions based on the amount entered.
   Write logic to determine whether the amount is positive, negative, or zero.
   Logic:
   - Get the amount from the user
   - Use if-elif statements to compare the amount
   - If greater than 0, print "Positive"
   - If less than 0, print "Negative"
   - Else, print "Zero"

2. **Scenario:** A digital locker requires users to enter a numerical passcode. As part of a security feature, the system checks the sum of the digits of the passcode.
   Write logic to compute the sum of the digits of a given number.
   Logic:
   - Get the number from the user
   - Initialize sum as 0
   - Use a loop to extract each digit using modulo (%) and add it to sum
   - Divide the number by 10 in each iteration
   - Print the final sum

3. **Scenario:** A mobile payment app uses a simple checksum validation where reversing a transaction ID helps detect fraud.
   Write logic to take a number and return its reverse.
   Logic:
   - Get the number from the user
   - Initialize a variable for reverse as 0
   - Use a loop to extract digits and build the reverse
   - In each loop, use modulo and integer division
   - Print the reversed number

4. **Scenario:** In a secure login system, certain features are enabled only for users with prime-numbered user IDs.
 Write logic to check if a given number is prime.
   Logic:
   - Get the number from the user
   - If number is less than 2, not prime
   - Loop from 2 to number - 1
   - If number is divisible by any, print “Not Prime”
   - Else, print “Prime”

5. **Scenario:** A scientist is working on permutations and needs to calculate the factorial of numbers frequently.
 Write logic to find the factorial of a given number using recursion.
   Logic:
   - Define a function that calls itself (recursion)
   - Base case: if number is 0 or 1, return 1
   - Recursive case: multiply number with factorial of (number-1)
   - Call the function and print result

6. **Scenario:** A unique lottery system assigns ticket numbers where only Armstrong numbers win the jackpot.
 Write logic to check whether a given number is an Armstrong number.
   Logic:
   - Get the number from the user
   - Count the number of digits
   - For each digit, raise it to the power of total digits and add to sum
   - If sum equals original number, it's Armstrong

7. **Scenario:** A password manager needs to strengthen weak passwords by swapping the first and last characters of user-generated passwords.
 Write logic to perform this operation on a given string.
   Logic:

- Get the string from the user
- If string length is less than 2, keep as it is
- Else, swap first and last characters
- Print the modified string

8. **Scenario:** A low-level networking application requires decimal numbers to be converted into binary format before transmission.
   Write logic to convert a given decimal number into its binary equivalent.
   Logic:
   - Get the decimal number from the user
   - Use built-in bin() function to convert
   - Remove ‘0b’ prefix and print

9. **Scenario:** A text-processing tool helps summarize articles by identifying the most significant words.
   Write logic to find the longest word in a sentence.
   Logic:
   - Get the sentence from the user
   - Split the sentence into words
   - Loop through the words and track the longest
   - Print the longest word

10. **Scenario:** A plagiarism detection tool compares words from different documents and checks if they are anagrams (same characters but different order).
    Write logic to check whether two given strings are anagrams.

    Logic:
    - Get two strings from the user
    - Remove spaces and convert to lowercase
    - Sort both strings
    - If sorted versions are same, they are anagrams