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**:

   1. Get transaction amount
   2. Check is amount is greater than 0 , If yes print positive
   3. If the amount is lesser then 0 print negative
   4. Else print the amount is 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:**
   1. Get the numerical passcode
   2. Take each digit in the passcode
   3. Add digits of the passcode
   4. Print the sum of digits

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:**
   1. Get the ID number
   2. Convert it to a string
   3. Reverse the string
   4. Again convert it to integer
   5. 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:**
   1. Get the ID number
   2. If the number is less than 2 print “not prime”
   3. Loop from 2 to the square root of the given number
   4. If it is divisible by any of the numbers it is not prime
   5. If it is not divisible by any of the numbers it is 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:**
   1. Read the input number
   2. Set factorial as 1
   3. Loop from 2 to then input number
   4. Multiple the number with factorial of previous number
   5. Print the 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:**
   1. Get the input number
   2. Check number of digits in the number(n)
   3. Take individual digits and raise to the power of n
   4. Take the sum of power of digits
   5. If the sum is equal to the input number its armstrong number
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:**
   1. Get the user password
   2. If length of string is greater than 2
   3. New password = password[-1]+password[1:-1]+passowrd[0]
   4. Print new password
   5. Otherwise print the same password

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:**
   1. Read the input decimal number
   2. Create an empty string to store binary values
   3. If number is greater than zero:
      - Divide the number by 2
      - Add the remainder to binary string
      - Divide the number by 2 using floor division
   4. Reverse the binary string
   5. Print the binary representation

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:**
    1. Get the input sentence
    2. Convert the sentence into individual words
    3. Create a variable to store the longest word
    4. Loop through each word
    5. If length of the new word is larger than the previous word
    6. Update the longest word
    7. 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:**

    1. Get the two string inputs
    2. Convert both strings into lowercase
    3. Sort both the strings
    4. If both the sorted strings are equal it an anagram
    5. Else not an anagram