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.

   I. Get the transactions from the user as int input
   II. Check the transactions is more than 0, positive
   III. Elif less than 0, negative
   IV. Else, 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.

   I. Get the passcode as int input
   II. Assign a variable to 0
   III. Each digit in the passcode will sum into the variable until the while loop ends

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.

   I. Get the transaction id as input
   II. Assign a reverse variable to 0
   III. Extract the last digit and store in reverse with loop function
   IV. Show the reverse ID

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.

   I. Get the User id as input
   II. If less than 2, not prime
   III. Check divisible by 2 to n**0.5, not prime

IV. Else, 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.

I. Get the factorial number as input
II. If 0 or 1, return 1
III. Else, n*factorial(n-1)

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.

I. Get the number as input
II. Count the length of digits
III. Assign a variable to 0
IV. Extract each digit
V. Raise digit to power of digit count
VI. Add them
VII. Compare with original number, if the sum = original, armstrong
VIII. Else, not an 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.

I. Take the password string
II. If length ≤ 1 → no change
III. Swap first and last characters
IV. Keep the middle part unchanged

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.

I. Get the decimal number

II. If the num is 0, then print it 0

III. Num>0, divide the number by 2 and store the remainder

IV. Build binary string by adding remainders (last to first)

V. Stop when number becomes 0

VI. Reverse the collected bits for final binary

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.

I. Get the sentence as input

II. Split sentence into words

III. Track the longest word seen so far

IV. Compare lengths of each word

V. Return the longest word found

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.

I. Get the two strings as input

II. Remove spaces and convert to lowercase

III. Sort both strings alphabetically

IV. If sorted strings are equal, they are anagrams

V. Else, not anagrams