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) **Read the numbers**
2) **If the number is greater than 0, print "Positive"**
3) **Else the number is less than 0, print "Negative"**
4) **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:

1. **Read the input number.**
2. **Convert the number into individual digits.**
3. **Initialize a sum variable to 0.**
4. **For each digit in the number. Add it to the sum variable.**
5. **Print 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. **Read the input number.**
2. **Convert the number to string.**
3. **Reverse the string.**
4. **Convert it back to a number.**
5. **Print "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. **Read the input number.**
2. **If the number is less than 2. Print "Not Prime"**
3. **Loop from 2 to the square root of the number:**
4. **If the number is divisible by any of theses values, print "Not Prime"**
5. **If no divisors found 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:

1. **Read the input number.**
2. **If the number is 0 or 1, return 1.**
3. **Else, return the number multiplied by the factorial of (number - 1).**
4. **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) **Read the input number.**
2) **Count the number of digits.**
3) **Initialize a sum variable to 0.**
4) **For each digit in the number:**
    a. **Raise the digit to the power of the total number of digits.**
    b. **Add the result to the sum variable.**
5) **If the sum is equal to the original number, print "Armstrong Number".**
6) **Else, print "Not an 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. **Read the input string.**
2. **If the string length is less than 2, print the string as is.**
3. **Swap the first and last characters while keeping the middle part unchanged.**
4. **Print the modified string.**

HOT AI

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 decimal number.**
2. **Initialize an empty string for binary representation.**
3. **While the number is greater than 0:**
    - **Divide the number by 2 and store the remainder.**
    - **Add the remainder to the binary string.**
    - **Update the number by dividing it by 2.**
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. **Read the input sentence.**
2. **Split the sentence into individual words.**
3. **Initialize a variable to store the longest word.**
4. **Loop through each word:**
5. **if the current word is longer than the stored longest word, update the longest word.**
6. **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. **Read the two input strings.**
2. **Remove spaces and convert both strings to lowercase.**
3. **Sort the characters of both strings.**
4. **If the sorted versions of both strings are identical, print "Anagram".**
5. **Else, print "Not an Anagram".**

HOPE AI