

**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.

**Write logic:**

- Read the input number.
- If the number is greater than 0, print "Positive (Deposit)".
- Else if the number is less than 0, print "Negative (Withdrawal)".
- Else, print "Zero (No Transaction)".

**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

**Write logic:**

- Read the input number.
- Convert the number into individual digits.
- Initialize a sum variable to 0.
- For each digit in the number, add it to the sum variable.
- Print the sum of the 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.

**Write logic:**

- Read the input number.
- Convert the number into a string.
- Reverse the string.
- Convert it back to a number.
- 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.

**Write logic:**

- Read the input number.
- If the number is less than 2, print "Not Prime".
- Loop from 2 to the square root of the number:
  - If the number is divisible by any of these values, print "Not Prime" and exit.
- If no divisors are 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.

**Write logic:**

- Read the input number.
- If the number is 0 or 1, return 1.
- Else, return the number multiplied by the factorial of (number - 1).
- 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.

**Write logic:**

- Read the input number.
- Count the number of digits.
- Initialize a sum variable to 0.
- For each digit in the number:
  - Raise the digit to the power of the total number of digits.
  - Add the result to the sum variable.
- If the sum is equal to the original number, print "Armstrong Number".
- 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.

**Write logic:**

- Read the input string.
- If the string length is less than 2, print the string as is.
- Swap the first and last characters while keeping the middle part unchanged.
- 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.

**Write logic:**

- Read the input decimal number.
- Initialize an empty string for binary representation.
- 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.
- Reverse the binary string.
- 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.

**Write logic:**

- Read the input sentence.
- Split the sentence into individual words.
- Initialize a variable to store the longest word.
- Loop through each word:
  - if the current word is longer than the stored longest word, update the longest word.
- 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

**Write logic:**

- Read the two input strings.
- Remove spaces and convert both strings to lowercase.
- Sort the characters of both strings.
- If the sorted versions of both strings are identical, print "Anagram".

Else, print "Not an Anagram"