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.

**Ramishahope Artificial Intelligence Pvt Ltd**

**36, Old Anandas, SG Arcade, Marudhamalai Main Road, Vadavalli, Coimbatore -641041.**

**+91 6385383227 | [www.hopelearning.net](http://www.hopelearning.net/) | [mdaravind@hopelearning.net](mailto:mdaravind@hopelearning.net) | 33AAMCR3722R1ZU**

**Logic:**

* 1. Read the amount entered
  2. Print “Positive” if amount is greater than 0
  3. Print “Negative: if amount is less than 0
  4. Print “Zero” otherwise

1. **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 numerical passcode
  2. Store the passcode in new variable i.e passcode
  3. Initialize sum as 0
  4. Check passcode greater than 0, repeat the steps e,f,g if Yes
  5. extract each digit (passcode%10)
  6. reduce the passcode (passcode//10)
  7. add the digit with existing sum
  8. otherwise print the sum

1. **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 numerical transaction id
  2. Convert the transaction id to string
  3. Reverse the string and convert it back to number
  4. Print the reversed transaction id

1. **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 number
  2. Check if number is less than 2, Print “Not Prime”
  3. Iterate loop from 2 to square root of the number
  4. If number is evenly divisible by any these values(3…square root(number) , exit the loop and Print “Not Prime”
  5. Print “Prime” if factors not found

1. **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 number
  2. check the termination condition as if number is equal to 1 return 1
  3. otherwise return the value of number \* factorial(number-1)

1. **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 number
  2. Find the total number of digits
  3. Initialize sum as 0
  4. Iterate each digit in the number
  5. Raise the digit to the power of total digits
  6. Add the raised value to the sum
  7. After Iteration exists, check sum is equal to the original number entered
  8. Print “Armstrong Number” if yes
  9. Print “Not a Armstrong Number” otherwise

1. **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 password string
  2. Do nothing if string length is less than 2
  3. Otherwise swap the first and last character of the string
  4. Print the updated string

1. **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 empty binary string
  3. Divide the decimal number by 2 and concatenate the remainder in the binary string
  4. Update the number dividing by 2
  5. Repeat the steps c and d while number greater than 0
  6. Reverse the binary string
  7. Print the binary string

1. **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 sentence
  2. Split the sentence into word lists
  3. Initialize a variable to store longest word (longest\_word) with empty
  4. Iterate each word in the word list
  5. If length of the word is greater than the longest word stored, update the longest word
  6. Repeat d and e step till end of iteration
  7. Print the longest word

1. **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 strings string1,string2
  2. Strip the spaces and convert both strings to lowercase
  3. Sort the characters of the two strings
  4. If sorted version of strings are same , then print “Anagram”
  5. Otherwise print “Not anagram”

**Ramishahope Artificial Intelligence Pvt Ltd**

**36, Old Anandas, SG Arcade, Marudhamalai Main Road, Vadavalli, Coimbatore -641041.**

**+91 6385383227 | [www.hopelearning.net](http://www.hopelearning.net/) | [mdaravind@hopelearning.net](mailto:mdaravind@hopelearning.net) | 33AAMCR3722R1ZU**