**Question 1: Write a program to count the number of vowels and consonants present in an input string.**

**Algorithm**:

Step 1:

Define the function ‘count\_vowels\_and\_consonants\_in\_word(word)’ which takes a word as am input.

Step 2:

Initialize the variable ‘vowels’ with the string containing vowels in both upper and lower case.

Step 3:

Count the number of vowels in the input word using a generator expression and store it in the variable ‘vowel\_count’.

Step 4:

Count the number of consonants in the input word using a generator expression and store it in the variable ‘consonant\_count’.

Step 5:

Create lists ‘vowels\_in\_word’ and ‘consonants\_in\_word’ containing the individual vowels and consonants in the word.

Step 6:

Return the counts and lists as tuple ‘(vowel\_count, consonant\_count, vowels\_in\_word, consonants\_in\_word)’.

**PseudoCode:**

Function count\_vowels\_and\_consonants\_in\_word(word):

vowels = “aeiouAEIOU”

vowel\_count=sum(1 for each char in word if char in vowels)

consonant\_count=sum(1 for char in word if chat.isalpha() and char not in vowels)

vowels\_in\_word = [char for char in word if char in vowels]

consonants\_in\_word = [char for char in word if char.isalpha() and char not in vowels]

return vowel\_count, consonant\_count, vowels\_in\_word, consonants\_in\_word

# Get user input

user\_word = input("Enter a word: ")

vowel\_count, consonant\_count, vowels, consonants = count\_vowels\_and\_consonants\_in\_word(user\_word)

result\_output={

"Word": user\_word,

"Vowel Count": vowel\_count,

"Consonant Count": consonant\_count,

"Vowels": vowels,

"Consonants": consonants

}

print(result\_output)

**Question 2: If the program that accepts two matrices A and B as input and returns their product AB. Check if A and B are multipliable; if not, return error message.**

**Algorithm:**

Step 1:

Define the matrix multiplication function “matrix\_multiply(matrix\_a, matrix\_b**)”.**

Step 2:

Get the dimensions of matrices A and B (rows and columns).

Step 3:

Check if the number of columns in matrix A is equal to the number of rows in matrix B for compatibility.

Step 4:

If matrices A and B are compatible, initialize the result matrix with zeros.

Step 5:

Use nested loops to perform matrix multiplication and populate the result matrix.

Step 6:

Return the result matrix.

Step 7:

Define the function get\_matrix\_from\_user(rows, cols, matrix\_name) to get a matrix from the user.

Step 8:

Take user input for the number of rows and columns for matrices A and B.

Step 9:

Use the get\_matrix\_from\_user function to get matrices A and B from the user.

Step 10:

Call the matrix\_multiply function with matrices A and B to get the result matrix.

Step 11:

Print the result matrix or an error message if the matrices are not compatible.

**Algorithm**

function matrix\_multiply(matrix\_a, matrix\_b):

rows\_a, cols\_a = len(matrix\_a), len(matrix\_a[0])

rows\_b, cols\_b = len(matrix\_b), len(matrix\_b[0])

if cols\_a != rows\_b:

return "Error: Matrices A and B are not multipliable."

result\_matrix = [[0 for \_ in range(cols\_b)] for \_ in range(rows\_a)]

for i in range(rows\_a):

for j in range(cols\_b):

for k in range(cols\_a):

result\_matrix[i][j] += matrix\_a[i][k] \* matrix\_b[k][j]

return result\_matrix

function get\_matrix\_from\_user(rows, cols, matrix\_name):

print(f"Enter elements for Matrix {matrix\_name}:")

return [[int(input(f"Enter element for Matrix {matrix\_name}[{i}][{j}]: ")) for j in range(cols)] for i in range(rows)]

matrix\_a\_rows = int(input("Enter the number of rows for Matrix A: "))

matrix\_a\_cols = int(input("Enter the number of columns for Matrix A: "))

matrix\_a = get\_matrix\_from\_user(matrix\_a\_rows, matrix\_a\_cols, 'A')

matrix\_b\_rows = int(input("Enter the number of rows for Matrix B: "))

matrix\_b\_cols = int(input("Enter the number of columns for Matrix B: "))

matrix\_b = get\_matrix\_from\_user(matrix\_b\_rows, matrix\_b\_cols, 'B')

result\_matrix = matrix\_multiply(matrix\_a, matrix\_b)

print("Matrix Multiplication Result:")

if isinstance(result\_matrix, str):

print(result\_matrix)

else:

for row in result\_matrix:

print(row)

**Question 3: Write a program to find the number of common elements between two lists. The lists contains intergers.**

**Algorithm:**

Step 1:

Define the function count\_common\_elements(list\_a, list\_b) to count the common elements between two lists.

Step 2:

Use the set intersection operation to find the common elements between list\_a and list\_b.

Step 3:

Return the count of common elements.

Step 4:

Define the function get\_integer\_list\_from\_user() to get a list of integers from the user.

Step 5:

Take user input for elements as a space-separated string, convert it to a list of integers, and return the list.

Step 6:

Get user input for two lists using the get\_integer\_list\_from\_user function.

Step 7:

Call the count\_common\_elements function with the two lists to get the count of common elements.

Step 8:

Print the result.

**Pseudo Code:**

function count\_common\_elements(list\_a, list\_b):

common\_elements = set(list\_a) & set(list\_b)

return len(common\_elements)

function get\_integer\_list\_from\_user():

return [int(x) for x in input("Enter elements (space-separated): ").split()]

list\_a = get\_integer\_list\_from\_user()

list\_b = get\_integer\_list\_from\_user()

common\_count = count\_common\_elements(list\_a, list\_b)

print(f"Number of Common Elements: {common\_count}")

**Question 4: Write a program that accepts a matrix as input and returns its transpose.**

**Algorithm:**

Step 1:

Define the function transpose\_matrix(input\_matrix) to transpose the given matrix.

Step 2:

Use list comprehension to create a new matrix where rows become columns and vice versa.

Step 3:

Get user input for the number of rows and columns in the matrix.

Step 4:

Use nested list comprehension to get user input for each element of the matrix.

Step 5:

Call the transpose\_matrix function with the user-input matrix to get the transposed matrix.

Step 6:

Print the original matrix and the transposed matrix.

**PseudoCode**

function transpose\_matrix(input\_matrix):

return [[input\_matrix[j][i] for j in range(len(input\_matrix))] for i in range(len(input\_matrix[0]))]

num\_rows = int(input("Enter the number of rows for the matrix: "))

num\_cols = int(input("Enter the number of columns for the matrix: "))

matrix = [[int(input(f"Enter element for matrix[{i}][{j}]: ")) for j in range(num\_cols)] for i in range(num\_rows)]

transposed\_matrix = transpose\_matrix(matrix)

print("Original Matrix:")

for row in matrix:

print(row)

print("\nTransposed Matrix:")

for row in transposed\_matrix:

print(row)