**MACHINE LEARNING (22AIE213)**

**ASSIGMENT-1**

**NAME: BANDI HARSHITHA REDDY**

**REG NO: AIE22108**

Q1: Write a program to count the number of vowels and consonants present in an input string.

**Pseudo Code:**

function count\_vowels\_and\_consonants(input\_word):

vowels = "aeiouAEIOU"

consonants\_count = 0

vowels\_count = 0

for char in input\_word:

if char.isalpha():

if char in vowels:

vowels\_count += 1

else:

consonants\_count += 1

return vowels\_count, consonants\_count

function main():

user\_input = get\_user\_input("Enter a word: ")

vowels, consonants = count\_vowels\_and\_consonants(user\_input)

print(f"Number of vowels in the given word: {vowels}")

print(f"Number of consonants in the given word: {consonants}")

if \_\_name\_\_=="\_\_main\_\_":

call main()

**Explanation:**

Function count\_vowels\_and\_consonants:

1. Stores all vowels: A string containing all vowels (uppercase and lowercase) is saved in vowels for easy checking.
2. Starts counting: Two counters, vowels\_count and consonants\_count, are set to 0 to track vowel and consonant occurrences.
3. Checks each letter: The code loops through each character in the input word:
   * Only letters matter: It checks if the character is an alphabet (e.g., 'a', 'B'). If not, it skips it.
   * Is it a vowel?: If the character is found in the vowels string, it's counted as a vowel.
   * Otherwise, consonant: If not a vowel, it's considered a consonant and counted.
4. Returns the results: After processing all characters, the function returns two values: the number of vowels and consonants found.

Function main:

1. Gets user input: It prompts the user to enter a word and stores it.
2. Counts vowels and consonants: It calls the count\_vowels\_and\_consonants function with the user's word and stores the returned counts.
3. Prints the results: Finally, it uses formatted strings to display the number of vowels and consonants found in the user's word.

Q2: 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.

**Pseudo Code:**

function matrix\_multiply(matrixA, matrixB):

rowsA, colsA = get\_dimensions(matrixA)

rowsB, colsB = get\_dimensions(matrixB)

if colsA != rowsB:

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

result\_matrix = initialize\_matrix(rowsA, colsB, 0)

for i in range(rowsA):

for j in range(colsB):

for k in range(colsB):

result\_matrix[i][j] += matrixA[i][k] \* matrixB[k][j]

return result\_matrix

function user\_input(matrix\_name):

rows = get\_user\_input("Enter the number of rows for Matrix " + matrix\_name)

cols = get\_user\_input("Enter the number of columns for Matrix " + matrix\_name)

elements = []

for i in range(rows):

row\_elements = []

for j in range(cols):

element = get\_user\_input(f"Enter element for Matrix {matrix\_name}[{i}][{j}]: ")

add element to row\_elements

add row\_elements to elements

return elements

function main():

matrixA = user\_input('A')

matrixB = user\_input('B')

result\_matrix = matrix\_multiply(matrixA, matrixB)

print("Matrix Multiplication Result:")

if is\_string(result\_matrix):

print(result\_matrix)

else:

for row in result\_matrix:

print(row)

if \_\_name\_\_ == "\_\_main\_\_":

call main()

**Explanation:**

Functions:

1. matrix\_multiply:
   * Checks if matrices have compatible dimensions for multiplication.
   * Initializes a result matrix with zeros.
   * Uses nested loops to calculate each element of the result matrix:
     + Loops through rows of the first matrix.
     + Loops through columns of the second matrix.
     + For each element in the result, sums the product of corresponding elements from each row of the first and column of the second matrix.
   * Returns the resulting product matrix.
2. user\_input:
   * Prompts the user for matrix dimensions (rows and columns).
   * Prompts the user for each element of the matrix, row by row and column by column.
   * Returns a list of lists representing the entered matrix.
3. main:
   * Gets user input for two matrices, A and B, using the user\_input function.
   * Calls matrix\_multiply to get the product matrix.
   * Prints the result matrix:
     + If an error message is returned, prints it directly.
     + Otherwise, iterates through each row of the matrix and prints them individually.

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

**Pseudo Code:**

function count\_common\_elements(list1, list2):

common\_elements = set(list1) & set(list2)

return len(common\_elements)

function main():

list1 = [int(ele) for ele in input("Enter elements for the first list (comma-separated): ").split('')]

list2 = [int(ele) for ele in input("Enter elements for the second list (comma-separated): ").split('')]

common\_elements\_count = count\_common\_elements(list1, list2)

print(f"Number of common elements: {common\_elements\_count}")

if \_\_name\_\_ == "\_\_main\_\_":

call main()

**Explanation:**

Function count\_common\_elements:

1. Converts to sets: It efficiently transforms both lists (list1 and list2) into sets using set(). Sets remove duplicates and allow for faster comparisons.
2. Finds common elements: The & operator performs a set intersection, keeping only elements that exist in both sets. This stores the common elements in a new set called common\_elements.
3. Counts elements: It uses the len() function to find the number of elements in the common\_elements set, representing the count of common elements between the original lists.
4. Returns the count: The function returns the calculated count of common elements.

Function main:

1. Gets user input: It prompts the user to enter two comma-separated lists of numbers and stores them in list1 and list2. The int() function ensures elements are converted to integers for numerical comparisons.
2. Calls counting function: It passes both lists to the count\_common\_elements function and stores the returned count in common\_elements\_count.
3. Prints the result: It uses an f-string to print a message displaying the number of common elements found.

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

**Pseudo Code:**

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]))]

function main():

num\_rows = get\_user\_input("Enter the number of rows for your matrix: ")

num\_cols = get\_user\_input("Enter the number of columns for your matrix: ")

matrix\_elements = []

for i in range(num\_rows):

row\_elements = []

for j in range(num\_cols):

element = get\_user\_input(f"Enter element for matrix[{i}][{j}]: ")

add element to row\_elements

add row\_elements to matrix\_elements

transposed\_matrix = transpose\_matrix(matrix\_elements)

print("Original Matrix:")

for row in matrix\_elements:

print(row)

print("\nTransposed Matrix:")

for row in transposed\_matrix:

print(row)

if \_\_name\_\_ == "\_\_main\_\_":

call main()

**Explanation:**

Function transpose\_matrix:

1. Iterates through columns: This function takes a matrix as input and loops through each column (j) of the matrix.
2. Creates new row: For each column, it creates a new empty list to represent a row in the transposed matrix.
3. Fills new row: Within the column loop, it iterates through each row (i) of the original matrix. For each row, it accesses the element at [j][i] (column j, row i) and adds it to the new row being created.
4. Returns the result: After processing all columns and creating new rows, the function returns a new matrix (transposed\_matrix) containing the transposed elements.

Function main:

1. Gets matrix dimensions: This function first asks the user to enter the number of rows and columns for the matrix.
2. Creates the matrix: It initializes an empty list matrix\_elements to store the matrix elements.
3. Gets element values: It uses nested loops to prompt the user for each element of the matrix, row by row and column by column, and adds them to matrix\_elements.
4. Performs transposition: It calls the transpose\_matrix function with the created matrix and stores the transposed version in transposed\_matrix.
5. Prints the matrices: Finally, it prints both the original and transposed matrices:
   * It iterates through each row of the original and transposed matrices and prints them individually.