**MACHINE LEARNING ASSIGNMENT 1**

(22AIE213)

**TEAM MEMBERS:**

SAI VIVEK B – [22107]

HARSHA VARDHAN TALUPULA – [22159]

ABHISHEK KUMAR K – [22120]

**SET – odd ( AIE22107)-B Sai Vivek**

Q1) PSEUDO CODE

Function count (list, sum):

Calucalte the length of the list

Initialize count as 0

for i=0 to length(list)

for j=0 to I+1,length(list)

if list[i] + list[k] == sum then

count=count+1

return count

end function

input\_list= [2,7,4,1,3,6]

sum=10

output gives the no.of pairs with sum 10

**Explanation:**

Function takes a list and a sum as Arguments, then two nested loops to iterate they are first one for to take one number again the nested loop takes the next number of that and checks sum is 10 in the if conditional loop then the count value increments by 1 similarly for all possible pairs it checks and returns the count of pairs in the list where the numbers sum is 10. Finally, the input list, sum are given and the function is called with the values present and sum.

Q2) PSEUDO CODE

Function range(List)

If (m <= 3) Then

Return “Range Determination is not possible”

Else

Minimum = min(List)

Maximum= max(List)

Return Maximum-Minimum

list = [5, 8, 3, 2, 7]

result=range(list)

print the result

**Explanation:**

The range function checks the Range which is not should be less than 3 and in else part it calculate the minimum and maximum of the list and return maximum- minimum that prints the range the given list

Q3) PSEUDO CODE

Functionmatrix multiplication(matrix1,matrix2):

The logic makes itself multiply no of times we gave its power and return it

function matrixpower(matrix, n):

if n is 0:

return identity matrix of the same shape as matrix

else if n is positive:

result = matrix

for i from 1 to n-1:

result = matrixmultiplication(result, matrix)

return result

result = inverse

for i from 1 to m-1:

result = matrixmultiplication(result, inverse)

return result

**Explanation:**

In the matrix multiplication function it makes logic makes itself multiply no of times we gave its power and return it and in the function matrix power with respective matrix and if thee power is 0 then then it return its idententity matrix of sam size.we have to give input and power in the code and the n it will print the result matrix.

Q4) PSEUDO CODE

function WordOccurrence(word):

dict = {}

for character in word:

if character in dict:

dict[character] += 1

else:

dict[character] = 1

return dict

function MaximumRepeated(dict):

maximumValue = 0

for key, value in dict.items():

if value > maximumValue:

maximumValue = value

maximumKey = key

return maximumValue, maximumKey

word = input("Enter the word: ")

maximum\_count, maximum\_char = MaximumRepeated(WordOccurrence(word))

print(f"The maximum occurring character is '{maximum\_char}' with a frequency of {maximum\_count}")

**Explanation:**

The word occurence function defined empty dictionary to store the character counts. It then iterates through each character in the input string, incrementing the count for each character in the dictionary. After the iteration, it finds the character with the maximum count and returns both the character and its count. The main program takes the user's input, calls the "word occurence" function, and prints the character with the maximum occurrence and its count.

**SET – odd ( AIE22159)-Harsha vardhan**

Q1) PSEUDO CODE

Function Pairs\_with\_sum10(list, sum):

Initialize count as 0

for i=0 to length(list) – 1

for k = i+1 to length(list) – 1

if list[i] + list[k] == sum then

count=count+1

end if

end for

end for

return count

end function

input\_list= [2,7,4,1,3,6]

sum=10

output gives the no.of pairs with sum 10

**Explanation:**

Here the function takes a list and a sum as input, and returns the count of pairs in the list where the numbers sum is 10. The function uses two nested loops to iterate i.e., first loop is for to take one number again the nested loop takes the next number of that and checks sum is 10 in the if conditional loop then the count value increments by 1 similarly for all possible pairs it checks. Finally, the input\_list, sum are given and the function is called with the values present and sum. The output displays the numbers of pairs with sum equal to 10

Q2) PSEUDO CODE

Function Range\_List(List, m)

If (m <= 3) Then

Return “Range Determination is not possible”

Else

user\_input(List,m)

Minimum = min(List)

Maximum= max(List)

Return Maximum-Minimum

End if

End Function

Function user\_input(List,m)

For i=0 to m-1

Element = input(“enter the elements”)

List.append(element)

End for

Return list

End function

n= Input(“Enter number of elements in the List: “)

List= [ ]

Output “The Range Difference is : “ +Range\_List(List,n)

**Explanation:**

Here, in above code the “Range\_List” function checks that the range is less than or equal to three (3) if less than that means it a display a message, if it is greater than 3 then it calculates the range difference of maximum and minimum through min and max functions. The main program takes the input as number as elements as input, initializes the list, and then calls the “Range\_List” function to determine the range difference.

Q3) PSEUDO CODE

Function Matrixmultiplication (A, B)

Result\_matrix= [ ]

For i=0 to length(A) – 1

row= [ ]

For j = 0 to length(B[0]) -1

Element=0

For k=0 to length(A[0])-1

Element = element + A[i][k] \* B[k][j]

End For

row.Append(element)

End for

Result\_matrix.Append(row)

End For

Return result\_matrix

End Function

Function identity\_matrix(size)

Return [[1 if i == j else for j=0 to size-1]for i=0 to size-1]

End function

Function power\_of\_matrix(length(matrix))

If length(matrix) != length(matrix[0]) then

Raise ValueError(‘Enter square matrix only’)

End If

Result\_matrix = identity\_matrix(length(matrix))

current\_power = pow\_matrix

While current\_power > 0 DO

IF current\_power % 2 == 1 THEN

Result\_matrix = Matrixmultiplication(Result\_matrix, matrix)

End If

matrix = Matrixmultiplication(matrix, matrix)

current\_power = current\_power // 2

End while

Return Result\_matrix

End function

order\_of\_matrix = INPUT("Matrix order: ")

matrix = [[int(INPUT(f"Enter element ({i+1}, {j+1}): ")) for j = 0 to order\_of\_matrix - 1] for i = 0 to order\_of\_matrix - 1]

pow\_matrix = INPUT("Enter the power of matrix: ")

Result\_matrix = power\_of\_matrix(matrix, pow\_matrix)

FOR row in Result\_matrix DO

Print(row)

End for

**Explanation:**

Here, above the “Matrixmultiplication” function calculates the product the product of two matrices, the "identity\_matrix" function generates an identity matrix, and the "power\_of\_matrix" function calculates the power of a matrix using the divide and conquer method. The main program takes the order of the matrix, its elements, and the power value as input, and then calls the "power\_of\_matrix" function to compute the result, which is subsequently printed.

Q4) PSEUDO CODE

Function Max\_occurence(input\_string)

count = {}

For i in input\_string do

If i in count then

count[i] = count[i] + 1

Else

count[i] = 1

End if

End for

result = maximum value in the count dictionary

count\_result = count[result]

Return result, count\_result

End function

x = input("Enter the string: ")

print("Input string:", x)

result, count\_result = Max\_occurence(x)

print("The Maximum Occurring Character is '{result}' and its count is {count\_result}.")

**Explanation:**

Here, “Max\_occurence” function takes input string and initializes an empty dictionary to store the character counts. It then iterates through each character in the input string, incrementing the count for each character in the dictionary. After the iteration, it finds the character with the maximum count and returns both the character and its count. The main program takes the user's input, calls the "Max\_occurence" function, and prints the character with the maximum occurrence and its count.

**SET – EVEN ( AIE22120)-Abhishek kumar**

**Question 1: Pseudo code**

Algorithm for finding the number of vowels and consonants in a given strings.

string = INPUT("Enter string: ")

vowels = 0

vowel = "aeiouAEIOU"

FOR i in string DO

IF i in vowel THEN

vowels = vowels + 1

END IF

END FOR

consonents = length(string) - vowels

PRINT("Number of vowels are: " + vowels)

PRINT("Number of consonents are: " + consonents)

**Explaination:**

The code counts the number of vowels and consonants in a user-input string. The code prompts the user to enter a string, initializes variables to count vowels and consonants, and then loops through each character in the input string. If the character is a vowel, the vowel count is incremented, and the consonant count is calculated by subtracting the vowel count from the total length of the string. Finally, the code displays the count of the vowels and consonants.

**Question 2: Pseudo code**

Algorithm for multiplying the matrices:

FUNCTION multiply(A, B)

C = empty matrix with dimensions ROW\_A x COL\_B

FOR i = 0 to ROW\_A - 1 DO

FOR j = 0 to COL\_B - 1 DO

C[i][j] = 0

FOR k = 0 to COL\_A - 1 DO

C[i][j] += A[i][k] \* B[k][j]

END FOR

END FOR

END FOR

RETURN C

END FUNCTION

**Explanation:**

The "multiply" function takes two matrices A and B, their dimensions, and initializes the result matrix C with zeros. It then checks if the matrices can be multiplied, and uses nested loops to perform the matrix multiplication. The main program takes the user's input, calls the "multiply" function, and prints the resulting matrix.

**Question 3: Pseudo code**

Algorithm to find same values in the lists:

list1 = [] // Initialize an empty list to store values entered by the user

size = INPUT("Enter the size of the list: ")

FOR i = 0 to size - 1 DO // Loop to input values for 'list1'

values = INPUT("Enter a value: ")

list1.APPEND(values)

END FOR

PRINT(list1)

list2 = []

size2 = INPUT("Enter the size of the list: ")

FOR i = 0 to size2 - 1 DO // Loop to input values for 'list2'

values = INPUT("Enter a value: ")

list2.APPEND(values)

END FOR

PRINT(list2)

similar\_elements = []

FOR a in list1 DO // Nested loops to find common elements between 'list1' and 'list2'

FOR b in list2 DO

IF a == b THEN

similar\_elements.APPEND(a) // Append the common elements to 'similar\_elements'

END IF

END FOR

END FOR

PRINT("The similar elements are: " + similar\_elements) // Display the common elements found

**Explanation:**

The code first creates two lists, 'list1' and 'list2', by taking the size of the lists as input from the user and then populating the lists with the values entered by the user. After creating the lists, it finds and displays the common elements between 'list1' and 'list2' in a list named 'similar\_elements'. The common elements are then printed to the console. This code essentially performs the task of finding the common elements between two lists.

**Question 4: Pseudo code**

Algorithm for the Transposition of the matrix

FUNCTION calculate\_transpose\_of\_matrix(matrix)

result\_matrix = CREATE A MATRIX WITH DIMENSIONS SWAPPED FROM THE INPUT MATRIX

FOR i = 0 to LENGTH OF MATRIX DO

FOR j = 0 to LENGTH OF MATRIX[0] DO

result\_matrix[j][i] = matrix[i][j]

END FOR

END FOR

RETURN result\_matrix

END FUNCTION

Matrix = [[7, 8], [9, 10]]

transposed\_matrix = CALL calculate\_transpose\_of\_matrix FUNCTION WITH Matrix AS ARGUMENT

FOR EACH row IN transposed\_matrix DO

PRINT row

END FOR

**Explanation:-**

The code states a function named 'calculate\_transpose\_of\_matrix' that takes a matrix as input and returns its transpose. The transpose of a matrix is obtained by interchanging its rows and columns. The function initializes a result matrix with dimensions swapped and then uses nested loops to iterate through each element of the original matrix and assign it to the transposed matrix. The code then defines a matrix named 'Matrix' and calculates its transpose using the defined function. Finally, it displays the original and transposed matrices. This code essentially performs the task of finding the transpose of a matrix