

**22AIE213- MACHINE LEARNING**

**Lab Assignment-01**

**Submission by the team**

Nossam Sri Chakradhar – BL.EN. U4AIE22132

Gopa Pulastya – BL.EN. U4AIE22116

Rishi Anirudh K – BL.EN. U4AIE22139

of

the Semester-4

of

Artificial Intelligence

**SET – A**

**QUESTION-1:**

The algorithm takes a user-provided string as input and meticulously counts the vowels and consonants within it. It accomplishes this by employing distinct counters for each category and meticulously examining each character in the string, classifying and tallying them accordingly. The final vowel and consonant counts are then presented as the output.

**Pseudocode:**

1. SET count = 0
2. FOR i = 1 TO length OF list DO

FOR j = i + 1 TO length OF list DO

1. IF list[i] + list[j] = target\_sum THEN

INCREMENT count

1. RETURN count
2. SET list = [2, 7, 4, 1, 3, 6]
3. SET sum = 10
4. SET paircount = pairsum(list, sum)
5. OUTPUT "The number of pairs in the list", list, "with a sum of", sum, "is:", paircount
6. END

**QUESTION-2:**

This program calculates the range of a list of real numbers. The range is the difference between the largest and smallest values in the list. First, it checks if the list has at least three elements, as a range cannot be determined with less. If valid, it finds the minimum and maximum values in the list. Finally, it subtracts the minimum from the maximum to get the range and displays it along with the input list.

**Pseudocode:**

1. FUNCTION calculaterange(lst):

IF length OF lst < 3 THEN

RETURN "The range determination is not possible"

1. SET min\_val = the smallest value in lst

SET max\_val = the largest value in lst

RETURN max\_val - min\_val

1. FUNCTION main():

INPUT input\_str

SET numbers = a list of numbers parsed from input\_str

1. SET result = calculaterange(numbers)

OUTPUT "The range of the list", numbers, "is:", result

1. CALL main()
2. END

**QUESTION-3**

This program calculates the power of a square matrix. It first prompts the user for the matrix dimensions and elements, ensuring valid input. Then, it checks if the matrix is square. If so, it uses two helper functions: multiply for matrix multiplication and power to calculate the exponentiation. The power function iteratively multiplies the matrix by itself m-1 times, where m is the exponent entered by the user. Finally, the program displays the resulting matrix raised to the power of m.

**Pseudocode:**

1. FUNCTION multiply (A, B):

SET result = length OF A x length OF B[0]

1. FOR i = 1 TO length OF A DO

FOR j = 1 TO length OF B[0] DO

FOR k = 1 TO length OF B DO

SET result[i][j] = result[i][j] + A[i][k] \* B[k][j]

RETURN result

1. FUNCTION power (A, m):

IF A is not a square matrix THEN

RETURN "Input matrix should be a square matrix"

1. SET result = A

FOR \_ = 1 TO m - 1 DO

SET result = multiply (result, A)

RETURN result

1. FUNCTION getmatrix():

REPEAT UNTIL valid dimensions are entered:

GET rows from user

GET columns from user

1. CREATE EMPTY matrix with dimensions rows x columns

GET elements from user row-wise

RETURN matrix

1. SET matrix = getmatrix()

GET exponent from user

1. SET result = power(matrix, exponent)

OUTPUT "A raised to the power of", exponent, "is:"

OUTPUT result

1. END

**QUESTION-4:**

This program finds the character that appears most frequently in a given string. It works by first iterating through each character in the string. If the character is a letter, it converts it to lowercase to ensure case-insensitivity. It then keeps track of the occurrence of each character in a dictionary, similar to a counter. Finally, it identifies the character with the highest count in the dictionary and returns both the character and its frequency as a result.

**Pseudocode:**

1. SET alphabet\_count = store character counts
2. FOR each character IN the input string DO

IF the character is an alphabetic character, THEN

CONVERT the character to lowercase

INCREMENT the count of the character in alphabet\_count

IF the character is not already in alphabet\_count THEN

ADD the character to alphabet\_count with a count of 1

1. SET maxchar = alphabet\_count with the highest count

SET maxcount = max\_char in alphabet\_count

1. RETURN maxchar, maxcount
2. INPUT from the user
3. CALL highestoccurrence(input\_string)
4. OUTPUT gives the highest count in the alphabet and string
5. END

**SET - B**

**QUESTION-1:**

From the question we can see that the problem deals with the key steps involved in counting the Vowels and the consonants in the string which we input to the algorithm. This is done in the way that first we will be taking the user input and then we will be creating two variables one is the vowels\_count and consonants\_count then we will traverse to the input string using a for loop if we find the vowel then we update the count similar will be done with the consonants. Finally, we display the count of the vowels and the consonants.

**Pseudocode:**

1. FUNCTION count ()

SET vowels\_count = 0

SET consonants\_count = 0

SET string\_length = LENGTH OF String

1. INPUT String
2. FOR i=1 TO string\_length DO

IF character AT i IN string is a vowel, THEN

ADD 1 TO Vowel\_count

ELSE

ADD 1 TO consonants\_count

1. END

**QUESTION-2:**

From the question we understand that we have to take the input of two matrices and take the input of both the matrices and then check if the matrix multiplication can be performed if performed it should return the product of both the matrices and if it cannot be multiplied then it should show up an error message saying that the matrices cannot be multiplied. The proposed way is as follow we have two functions in which one function will be taking the user input and the other function checks the dimensions of the input matrices and performs the matrix multiplication by taking the input matrices if possible. Finally, then prints the result.

**Pseudocode:**

1. INPUT rows, cols
2. SET matrix = dimensions rows x cols
3. FOR i=1 TO rows DO

FOR j =1 TO cols DO

INPUT matrix[i][j]

1. REPEAT Step- 1, 2, 3 FOR another matrix
2. IF cols\_a != rows\_b THEN

OUTPUT “Error: Matrices cannot be multiplied”

1. SET result\_matrix [ rows\_a ] x [ cols\_b]
2. FOR i=1 TO rows\_a DO

FOR j=1 TO cols\_b DO

SET result\_matrix[i][j] =0

FOR k=1 TO cols\_a DO

ADD matrix\_a[i][j]\*matrix\_b[k][j] TO result\_matrix[i][j]

1. OUTPUT “The product of both matrices is AB”

FOR i=1 TO rows\_a DO

OUTPUT result\_matrix[i]

1. END

**QUESTION-3:**

From the question we can understand that we have to take the user input of the integers into two different lists and we have to compare the similar elements in the list and print the count of the similar elements and the similar elements. For this we have used two different functions one to take the input of the integers into the lists and other to find the common elements among the two lists and then we call two different functions and create an instance and use it to finally get the output result.

**Pseudocode:**

1. FUNCTION listinput():

WHILE TRUE DO

INPUT input\_str

TRY

SET integer\_list = EMPTY list

FOR i = 1 TO LENGTH OF input\_str DO

SPLIT input\_str by spaces INTO words

ADD integer(word) TO integer\_list

RETURN integer\_list

EXCEPT ValueError:

OUTPUT "Error: Only integer values are allowed."

1. FUNCTION common (list\_a, list\_b):

SET common\_elements = SET OF list\_a INTERSECT SET OF list\_b

RETURN length OF common\_elements

1. INPUT list\_a using listinput()

INPUT list\_b using listinput()

1. SET common\_count = common (list\_a, list\_b)
2. END

**QUESTION-4:**

From the question we can see that the we have to the user input of the matrix and then find the transpose of the matrix. We have used rows and cols to take the input of the matrix with respect to rows and columns and we will define a matrix and then we swap the rows and columns and print the transposed matrix.

**Pseudocode:**

1. INPUT rows, cols
2. SET matrix [rows] x [cols]
3. FOR i = 1 TO rows DO

FOR j = 1 TO cols DO

INPUT matrix[i][j]

1. SET transposed\_matrix [cols] x [rows]
2. FOR i = 1 TO rows DO

FOR j = 1 TO cols DO

SET transposed\_matrix[j][i] = matrix[i][j]

1. OUTPUT "Transposed Matrix:"

FOR i = 1 TO cols DO

OUTPUT transposed\_matrix[i]

1. END