# Machine Learning Assignment – 1

Date: 31/ 01/ 2024

Team:

Ishvarya G – BL.EN.U4AIE22118

Kavya Sree Kammari – BL.EN.U4AIE22121

Sharanya V Thambi – BL.EN.U4AIE22151

# Kavya Sree Kammari (Odd Set)

Overall, I’ve divided the Questions into multiple functions and taken inputs in the main function. Each code explains about the efficient method involved and time complexities.

## Question 1:

Question 1 aims to find the number of pairs in each list that sums up to a target value. The function utilizes a hash map to efficiently track visited numbers while iterating through the list giving a big o notation O(n).

Pseudocode:

```
Function count_Pairs_With_Sum(list, target_sum):
    Initialize no_of_pairs to 0
    Initialize list_for_visiting as an empty list
    For each number in list:
        potential_pair = target_sum - number
        If potential_pair is in list_for_visiting:
            Increment no_of_pairs
        Add number to list_for_visiting
Return no_of_pairs
```

## Question 2:

Function Question_2 calculates the range of a given list, which is the difference between its maximum and minimum values. The code efficiently determines the range of the list by iterating through its elements and keeping track of minimum and maximum values with the big O notation of O(n).

Pseudocode:

```
Function calculateRange(list):
    minimum_number = positive infinity
    maximum_number = negative infinity
    For each element in the list:
        If element < minimum_number:
            Update minimum_number with the element
        If element > maximum_number:
            Update maximum_number with the element
    range = maximum_number - minimum_number
    Return range
```

## Question 3:

Question 3 function raises a square matrix to a given exponent using matrix multiplication function defined in it. The function checks if the matrix is square, initializing with an identity matrix, and iteratively multiplies the matrix by itself for m times (exponents).

Pseudocode:

```
    Function matrix_exponentiation(matrix, exponent):
        size = size of the matrix
    If matrix is not square:
            Return "Non - square matrix"
        Define matrix_multiply(a, b) function:
        Return a matrix resulting from element-wise multiplication of
            matrices a and b
        Result_matrix = identity matrix
        Repeat exponent times:
            Update result matrix by multiplying it with the original matrix
            using matrix_multiply function
        return the final result_matrix
```

## Question 4:

Question 4 function is used to find the alphabet with the highest frequency in each string. It uses a default dict to store the frequency of each alphabet, filters out non – alphabetic character, and then determines the alphabet with the maximum occurence.

Pseudocode:

```
Function find_Max_Frequency_Alphabet(input_string):
    empty dictionary count_dict = defaultdict(int)
    For each character in the input_string:
        If the character is an alphabet:
            Increment the value of the corresponding key in count_dict
    max_char = max(count_dict, key = count_dict.get)
    Extract the maximum frequency value from count_dict
    return max_char, max_count
```

# Ishvarya G (Even Set)

In this program, I have made the code modularized specific to each question's functionality. The main function provides the functionality of a menu-driven system where the user is given a choice to select which functionality is required and based on that the corresponding input values are taken and a call is made to the respective function, this continues until the user enters "stop" to exit the program.

## Question 1:

*To find the number of Vowels and consonants given an input string :*

The code uses for loop to loop through each character in the string 's' and increments the count variables for keeping count of vowels and consonants accordingly.

*Pseudocode:*

```
    Initialise 2 lists 'vowels' and 'consonants'
Function VowConsCount( INPUT s):
        SET count_vowels=0
        SET count_consonants=0
FOR any character in s:
        Check if the character is in consonants:
                If so increment count_consonants by 1
        Check if the character is in vowels:
                If so increment count_vowels by 1
REPEAT until end of s
RETURN count_vowels and count_consonants
```

## Question 2:

*To find the product of 2 matrices and return an error if not multiplicable:*

The code defines a function that uses 3 for loops that loops through the range of rows of matrix 1 and columns of matrix 2 (size of product matrix) and finally through the range of rows of matrix 2 to find each element of the product matrix by taking the sum of the product of the elements in row 1 of matrix 1 and column 1 of matrix 2 and returns it. If in case the column of matrix 1 does not match the rows of matrix 2, it returns an error.

*Pseudocode :*

```
  Function matrixMult( INPUT mat1,INPUT mat2)
      rows_mat1,cols_mat1 = length of mat1, length of mat1[0]
      rows_mat2,cols_mat2 = length of mat2, length of mat2[0]

      IF cols_mat1 does not equal rows_mat2:
           RETURN “Error : Matrices A and B are not multiplicable”

      SET the result_matrix rows equal to mat1 and columns equal to mat2

       FOR i in range of length of rows_mat1:
              FOR j in range of length of cols_mat2:
                     FOR k in range of length of rows_mat2:
                            SET result_matrix[i][j] += mat1[i][k] * mat2[k][j]

      RETURN result_matrix
```

## Question 3:

*To find common elements in 2 lists of integers:*

The code uses list comprehension to store only numbers that are present in both the input lists into the new list that stores the common elements and returns it.

*Pseudocode :*

```
  Function commonElem ( Input list1, Input list2):
      common_elements = [number present in list1 only if the same number is present in list2]
      RETURN common_elements
```

## Question 4:

## *To find the transpose of a matrix:*

The code defines a function transpose that takes in an input matrix, ‘mat’. We define a new matrix ‘transposed’ and using 2 for loops we assign the elements along the rows of ‘mat’ to the columns of ‘transposed’.

## *Pseudocode:*

```
Function transpose( INPUT mat):
    SET transposed = 0s with number of rows equal to rows of mat and number of columns equal to columns of mat
    FOR I in range of length of mat:
        FOR j in range of length of mat[0]:
            SET transposed[j][i] =   mat[i][j]
RETURN transposed
```

# Sharanya Vanraj Thambi (Odd Set)

The program is created by having separate functions for each question and the main method allows the user to choose which question they desire to choose. This is done using a switch case built by if else conditions. Each program is done with the approach of iterating through the necessary elements of the list. The usage of extra variables has been minimised to save memory and all inputs are taken dynamically by the user.

## Question 1:

Consider the given list as [2, 7, 4, 1, 3, 6]. Write a program to count pairs of elements with sum equal to 10.

**Explanation:**

The program iterates through each element and compares each by further looping through all the elements, matching the selected number with each in the list, when the selected number and another number from the list adds up to 10 count is incremented by 1, count is then returned.

**Pseudo code:**

```
Function PairOfElements(list):
    Initialise sum to 0
    Initialse count to 0
    Loop i from 0 to 6:
        Loop j from i to 6:
            If((list1[i]+list1[j]) is equal to 10:
                Increment count by 1
    return count
```

## Question 2:

Write a program that takes a list of real numbers as input and returns the range (difference between minimum and maximum) of the list. Check for list being less than 3 elements in which case return an error message (Ex: “Range determination not possible”). Given a list [5,3,8,1,0,4], the range is 8 (8-0).

**Explanation:**

The program initialises max to zero because it is the least possible value above which anything is greater is not possible and initialises min to highest value possible so that all numbers are lesser than it, now all the numbers in the list are traversed through and if the numbers are lesser than min variable value then min is updated to that number. If the number is greater than that in max then the max variable is updated to that number. Then the range subtracting max and min is returned.

**Pseudo code:**

```
Function rangeFind(list2):
    Initialise max to -sys.maxsize
    Initialise min to sys.maxsize
    Loop I from 0 to length of list2
        if list2[i] is greater than equal to max:
            max=list2[i]
        if list2[i] is lesser than equal to max:
            min=list2[i]
    return (max-min)
```

## <u>Question 3:</u>

Write a program that accepts a square matrix A and a positive integer m as arguments and returns $A^m$.

**Explanation:**

The matrix is input in the main method after which it is passed to the function. Then by iterating through appropriate elements the computed elements of the multiplied matrix are appended to the new matrix this matrix is the passed through the function again in a loop iterating till m-1 present in the main section, where m is the power value input from user.

**Pseudo code:**

```
Function matrixmulti(matrix,n):
    Initialise a list multi to empty
    Initialise x to 0
    Loop interating i from 0 to length
        Initialse list row to empty
        Loop interating j from 0 to length
            Initialise x to 0
```

```
                    Loop interating k from 0 to length
                        Assign x to x+matrix[i][j]*matrix[k][j]
                    Append x to row
                Append row to multi
            return multi


        In #main
        Input the power to raise matrix to as m
        Iterate i till m-1:
            Assign matrix_input to matrixmulti(matrix_input, n)
```

## Question 4:

Write a program to count the highest occurring character & its occurrence count in an input string. Consider only alphabets. Ex: for “hippopotamus” as input string, the maximally occurring character is ‘p’ & occurrence count is 3.

**Explanation:**

The program loops through each letter of the word and then compares each letter by further looping through each letter and comparing the selected letter with all other letters and if a match is found it increments count by 1, if this count is greater than max_count which is initialised as zero in the beginning it is updated. This way the maximum count will be stored in max_count along with the corresponding stored in maxletter.

**Pseudo code:**

```
Function word_count(word):
    Declare list3 and initialise it to empty
     max_count to 0
    Loop through each character i in the word:
        Initialise count to 0
        Loop through each character j in word:
            if i= = j :
                Increment count by 1
        if count >= max_count
                Assign count to max_count
                Initialise maxletter to i
Return max_count and maxletter
```