COVER PAGE

Made By: Manvendra Gosain

**Q1. Matrix transpose wihtin the matrix**

matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9],

]

print("Original matrix:")

for row in matrix:

print(row)

print("\nTransposed matrix:")

for i in range(len(matrix)):

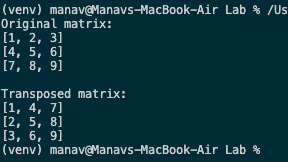
for j in range(i):

matrix[i][j], matrix[j][i] = matrix[j][i], matrix[i][j]

for row in matrix:

print(row)

**Output:**



**Q2. Matrix multiplication**

**def matrix\_multiplication(matrix1, matrix2):**

**n1, m1 = len(matrix1), len(matrix1[0])**

**n2, m2 = len(matrix2), len(matrix2[0])**

**# Check if the matrices can be multiplied**

**if m1 != n2:**

**raise ValueError("Matrices cannot be multiplied")**

**# Initialize the result matrix**

**result = [[0 for \_ in range(m2)] for \_ in range(n1)]**

**# Perform matrix multiplication**

**for i in range(n1):**

**for j in range(m2):**

**for k in range(m1):**

**result[i][j] += matrix1[i][k] \* matrix2[k][j]**

**return result**

**# Example usage**

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

**matrix1 = [**

**[1, 2],**

**[3, 4],**

**]**

**matrix2 = [**

**[1],**

**[2],**

**]**

**result = matrix\_multiplication(matrix1, matrix2)**

**for row in result:**

**print(row)**

**Output:**



**Q3. Gauss Elimination**

def gauss\_elimination(augmented\_matrix):

n = len(augmented\_matrix)

# Forward elimination

for i in range(n):

for j in range(i + 1, n):

factor = augmented\_matrix[j][i] / augmented\_matrix[i][i]

for k in range(i, n + 1):

augmented\_matrix[j][k] -= factor \* augmented\_matrix[i][k]

# Back substitution

solution = [0] \* n

for i in range(n - 1, -1, -1):

solution[i] = augmented\_matrix[i][n]

for j in range(i + 1, n):

solution[i] -= augmented\_matrix[i][j] \* solution[j]

solution[i] /= augmented\_matrix[i][i]

return solution

# Example usage

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

# Example augmented matrix

augmented\_matrix = [

[-1, -3, -1, 1, -7],

[1, -4, -3, -4, -3],

[1, 5, 2, 6, -3],

[10, 4, -2, -2, 6],

]

solution = gauss\_elimination(augmented\_matrix)

print("Solution:", solution)

**Output:**



**Q4. Gauss Jordan**

def gauss\_jordan(augmented\_matrix):

n = len(augmented\_matrix)

m = len(augmented\_matrix[0])

# Forward elimination

for i in range(n):

# Make the diagonal contain all 1s

divisor = augmented\_matrix[i][i]

for j in range(i, m):

augmented\_matrix[i][j] /= divisor

# Eliminate other rows

for j in range(n):

if j != i:

factor = augmented\_matrix[j][i]

for k in range(i, m):

augmented\_matrix[j][k] -= factor \* augmented\_matrix[i][k]

# Extracting the solution

solution = [augmented\_matrix[i][-1] for i in range(n)]

return solution

# Example usage

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

augmented\_matrix = [

[-1, -3, -1, 1, -7],

[1, -4, -3, -4, -3],

[1, 5, 2, 6, -3],

[10, 4, -2, -2, 6],

]

solution = gauss\_jordan(augmented\_matrix)

print("Solution:", solution)

**Output:**



**Q5. Matrix inversion**