# MAI LAB FILE

**By:**

**Name** **-** Shantanu Shukla

**Course -** M.Tech. AI & DS

**Enrollment No. -** 00111804824

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Problem Statement** | **PAGE No.** |
| 1. | Write an algorithm to multiply two matrices Amxn and Bnxp | 3-5 |
| 2. | Write an algorithm to transpose a square matrix A within A | 6-7 |
| 3. | Implement Gauss Elimination method | 8-9 |
| 4. | Implement Gauss Jordan method | 10-11 |
| 5. | Write a program to check whether the matrix is invertible or not, if yes find the inverse | 12-14 |

**EXERCISE-1**

**Problem Statement:** Write an algorithm to multiply two matrices Amxn and Bnxp.

# Function to take matrix input from user

def input\_matrix(name):

rows = int(input(f"Enter the number of rows for {name}: "))

cols = int(input(f"Enter the number of columns for {name}: "))

matrix = []

print(f"Enter the elements for {name} row-wise:")

for i in range(rows):

row = list(map(int, input(f"Row {i + 1}: ").split()))

matrix.append(row)

return matrix

# Function to multiply two matrices

def multiply\_matrices(matrix1, matrix2):

# Get dimensions

rows\_matrix1 = len(matrix1)

cols\_matrix1 = len(matrix1[0])

rows\_matrix2 = len(matrix2)

cols\_matrix2 = len(matrix2[0])

# Ensure multiplication is valid

if cols\_matrix1 != rows\_matrix2:

raise ValueError("Number of columns in Matrix 1 must equal number of rows in Matrix 2")

# Initialize the result matrix with zeros

result = [[0 for \_ in range(cols\_matrix2)] for \_ in range(rows\_matrix1)]

# Perform matrix multiplication

for i in range(rows\_matrix1):

for j in range(cols\_matrix2):

for k in range(cols\_matrix1):

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

return result

# Function to display matrix

def display\_matrix(matrix):

for row in matrix:

print(" ".join(map(str, row)))

# Main program

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

print("Matrix 1")

matrix1 = input\_matrix("Matrix 1")

print("\nMatrix 2")

matrix2 = input\_matrix("Matrix 2")

try:

result = multiply\_matrices(matrix1, matrix2)

print("\nResultant Matrix after multiplication:")

display\_matrix(result)

except ValueError as e:

print(f"Error: {e}")

**EXERCISE-2**

**Problem Statement:** Write an algorithm to transpose a square matrix A within A.

# for taking input from user for m\*n matrix

n = int(input("Enter the number of rows and columns in nxn matrix: "))

# for taking the matrix input by the user

matrix = []

for i in range(n):

print(f"Enter the {i+1} row of the matrix:")

a = []

for j in range(n):

a.append(int(input()))

matrix.append(a)

# printing the original matrix

print("The matrix is: ")

for i in range(n):

for j in range(n):

print(matrix[i][j], end=" ")

print()

# here is the fuction which transpose the matrix within the same matrix

for i in range(n):

for j in range(i+1):

temp=matrix[i][j]

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

matrix[j][i]=temp

# printing the transpose of the original matrix

print("The matrix after being transposedis: ")

for i in range(n):

for j in range(n):

print(matrix[i][j], end=" ")

print()

**EXERCISE-3**

**Problem Statement:** Implement Gauss Elimination method.

def gauss\_elimination():

n = int(input("Enter the number of unknowns: "))

a = []

# Reading the Augmented Matrix

print("Enter the Augmented Matrix row-wise:")

for i in range(n):

b = []

print(f"Enter the Row {i+1}:")

for j in range(n+1):

b.append(int(input()))

a.append(b)

print("\n")

# printing the matrix

print("The matrix is: ")

for i in range(n):

for j in range(n+1):

print(a[i][j], end=" ")

print()

# Applying Gauss Elimination

for i in range(n-1):

if a[i][i] == 0.0:

print("Mathematical error!")

return

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

c = a[j][i] / a[i][i]

for k in range(n+1):

a[j][k] = a[j][k] - c \* a[i][k]

# Back Substitution

x = [0 for \_ in range(n)]

x[n-1] = a[n-1][n] / a[n-1][n-1]

for i in range(n-2, -1, -1):

x[i] = a[i][n]

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

x[i] = x[i] - a[i][j] \* x[j]

x[i] = x[i] / a[i][i]

# Displaying the Solutions

print("\nSolutions:")

for i in range(n):

print(f"x[{i+1}] = {x[i]:.3f}")

# here we use the Gauss Elemination Function

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

gauss\_elimination()

**EXERCISE-4**

**Problem Statement:** Implement Gauss Jordan method.

def gauss\_jordan():

n = int(input("Enter the number of unknowns: "))

a = []

# Reading the Augmented Matrix

print("Enter the Augmented Matrix row-wise:")

for i in range(n):

b = []

print(f"Enter the Row {i+1}:")

for j in range(n+1):

b.append(int(input()))

a.append(b)

print("\n")

# printing the matrix

print("The matrix is: ")

for i in range(n):

for j in range(n+1):

print(a[i][j], end=" ")

print()

# Applying Gauss Jordan

for i in range(n):

if a[i][i] == 0.0:

print("Mathematical error!")

return

for j in range(n):

if i != j:

ratio = a[j][i]/a[i][i]

for k in range(n+1):

a[j][k] = a[j][k] - ratio \* a[i][k]

# printing the matrix

print("The matrix after applying Gauss Jordan elimination is: ")

for i in range(n):

for j in range(n+1):

print(a[i][j], end=" ")

print()

# printing the solution

print("The solution is: ")

for i in range(n):

print(f"x{i+1} = {a[i][n]/a[i][i]}")

# here we use the Gauss Jordan Function

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

gauss\_jordan()

**EXERCISE-5**

**Problem Statement:** Write a program to check whether the matrix is invertible or not, if yes find the inverse.

# for taking input from user

n = int(input("Enter the number of rows and columns in nxn matrix: "))

# for taking the matrix input by the user

matrix = []

for i in range(n):

print(f"Enter the {i+1} row of the matrix:")

a = []

for j in range(n):

a.append(int(input()))

matrix.append(a)

# printing the original matrix

print("The matrix is: ")

for i in range(n):

for j in range(n):

print(matrix[i][j], end=" ")

print()

# checking if the matrix is invertible or not

## here we get the minor of the matrix

def get\_minor(matrix, row, col):

return [r[:col] + r[col+1:] for r in (matrix[:row] + matrix[row+1:])]

## here we get the determinant of the matrix

def determinant(matrix):

if len(matrix) == 1: # Base case for 1x1 matrix

return matrix[0][0]

if len(matrix) == 2: # Base case for 2x2 matrix

return matrix[0][0] \* matrix[1][1] - matrix[0][1] \* matrix[1][0]

det = 0

for c in range(len(matrix)):

det += ((-1) \*\* c) \* matrix[0][c] \* determinant(get\_minor(matrix, 0, c))

return det

## here we chec if the matrix is invertible or not

if determinant(matrix) == 0:

print("The matrix is not invertible")

exit()

else:

print("The matrix is invertible")

# for getting the inverse of the matrix if the matrix is invertible

def inverse(matrix):

det = determinant(matrix)

for i in range(len(matrix)):

for j in range(len(matrix[0])):

matrix[i][j] =determinant(get\_minor(matrix, i, j))\*(-1)\*\*(i+j) / det

return matrix

matrix1 =inverse(matrix)

print("The inverse matrix is: ")

for i in range(len(matrix1)):

for j in range(len(matrix1[0])):

print(matrix1[i][j], end=" ")

print()