Assignment 2

Name: Ashish Nanda

Roll No.: J041

In [1]:

```
# Importing libraries
import numpy as np
import time
```

Task 1: Prove the properties of matrix multiplication

Matrix Creation

In [2]:

```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
A
```

Out[2]:

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [3]:

```
B = np.array([[10, 11, 12], [13, 14, 15], [16, 17, 18]])
B
```

Out[3]:

```
array([[10, 11, 12],
       [13, 14, 15],
       [16, 17, 18]])
```

In [4]:

```
C = np.array([[19, 20, 21], [22, 23, 24], [25, 26, 27]])
C
```

Out[4]:

```
array([[19, 20, 21],
       [22, 23, 24],
       [25, 26, 27]])
```

Property 1: Non-Commutative

A.B ≠ B.A

In [5]:

```
# A.B
AB = np.dot(A, B)
AB
```

Out[5]:

Out[5]:

```
array([[ 84,  90,  96],
       [201, 216, 231],
       [318, 342, 366]])
```

In [6]:

```
# B.A
BA = np.dot(B, A)
BA
```

Out[6]:

```
array([[138, 171, 204],
       [174, 216, 258],
       [210, 261, 312]])
```

Commutative only if either A or B is an identity matrix

A.I = I.A

In [7]:

```
# Matrix B = I
I = np.identity(3, dtype=int)
I
```

Out[7]:

```
array([[1, 0, 0],
       [0, 1, 0],
       [0, 0, 1]])
```

In [8]:

```
# A.I
m1 = np.dot(A, I)
m1
```

Out[8]:

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [9]:

```
# I.A
m2 = np.dot(I, A)
m2
```

Out[9]:

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

Property 2: Associative property

### A.(B.C) = (A.B).C

In [10]:

```
# A.(B.C)
A_BC = np.dot(A, np.dot(B, C))
A_BC
```

Out[10]:

```
array([[ 5976,  6246,  6516],
       [14346, 14994, 15642],
       [22716, 23742, 24768]])
```

In [11]:

```
# (A.B).C
AB_C = np.dot(np.dot(A, B), C)
AB_C
```

Out[11]:

```
array([[ 5976,  6246,  6516],
       [14346, 14994, 15642],
       [22716, 23742, 24768]])
```

Property 3: Distributive property

A.(B + C)= A.B + A.C

In [12]:

```
# A.(B + C)
m3 = np.dot(A, (B + C))
m3
```

Out[12]:

```
array([[222, 234, 246],
       [537, 567, 597],
       [852, 900, 948]])
```

In [13]:

```
# A.B + A.C
m4 = np.dot(A, B) + np.dot(A, C)
m4
```

Out[13]:

```
array([[222, 234, 246],
       [537, 567, 597],
       [852, 900, 948]])
```

Task 2: Calculate inverse of a matrix using Numpy

Random matrix creation

In [14]:

```
matrix = np.random.randint(100, size=(3, 3))
matrix
```

Out[14]:

```
array([[ 6, 93, 79],
       [15, 30, 48],
       [56, 61, 83]])
```

In [15]:

```
inv_matrix = np.linalg.inv(matrix)
inv_matrix
```

Out[15]:

```
array([[-0.00615722, -0.04076698,  0.02943657],
       [ 0.02028509, -0.05519006,  0.01260965],
       [-0.01075405,  0.0680668 , -0.01707996]])
```

Task 3: Comparison of Numpy and traditional looping

Random and empty matrix creation

In [16]:

```
matrix1 = np.random.randint(100, size=(10000, 10000))

matrix2 = np.empty((10000, 10000))
```

In [17]:

```
matrix1.shape, matrix2.shape
```

Out[17]:

```
((10000, 10000), (10000, 10000))
```

Traditional looping

In [18]:

```
initial = time.time()

for i in range(len(matrix1)):
    for j in range(len(matrix1)):
        matrix2[i][j] = matrix1[i][j] + 5

final = time.time()

print(f'Time taken using looping: {final - initial}')
```

```
Time taken using looping: 98.81207132339478
```

Using Numpy

In [19]:

```
initial = time.time()

matrix2 = np.add(matrix1, 5)

final = time.time()

print(f'Time taken using Numpy: {final - initial}')
```

```
Time taken using Numpy: 0.24549293518066406
```