CSCE 5150 – Analysis of Computer Algorithms

Programming Assignment 1 – Divide and Conquer

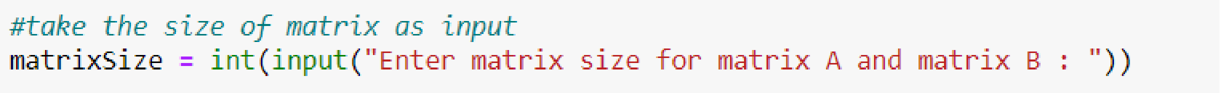
Neha Goud Baddam
11519516

Strassen's matrix multiplication Algorithm:

<u>Definition:</u> Strassen algorithm is a recursive method for matrix multiplication where we divide the matrix into 4 sub-matrices of dimensions n/2 x n/2 in each recursive step.

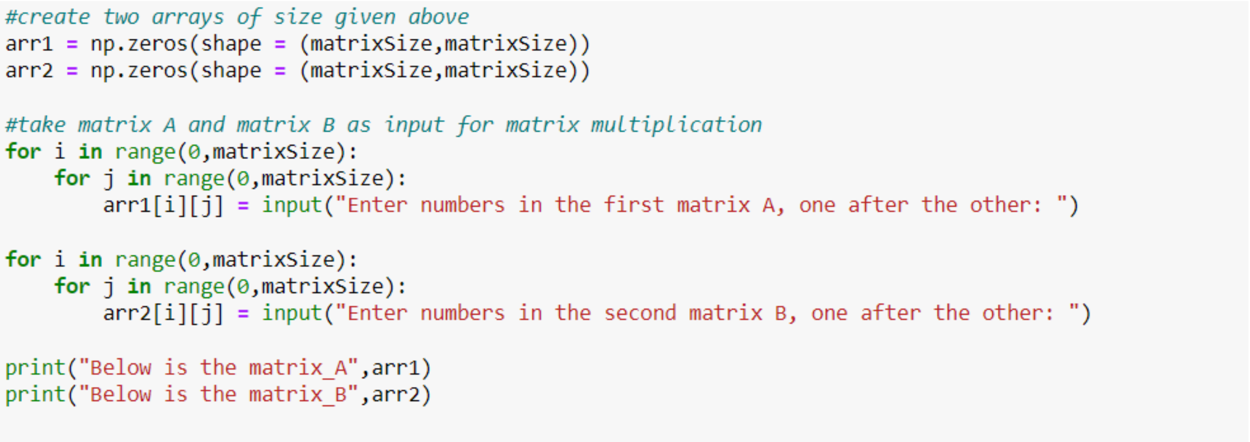
$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \times \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \begin{pmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{pmatrix}$$

1. Initially, the code will take the size of matrix A and matrix B as input.



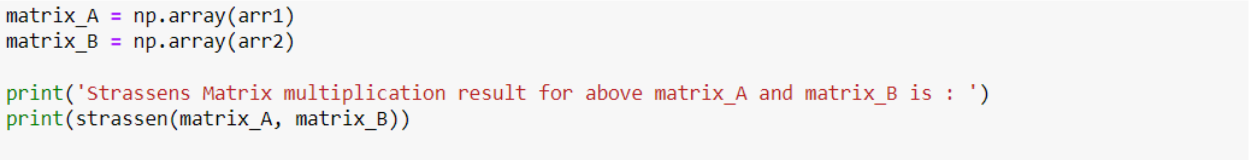
```
#take the size of matrix as input
matrixSize = int(input("Enter matrix size for matrix A and matrix B : "))
```

2. Now, I have created two multidimensional arrays, to take matrix A and matrix B as input. The numbers in the matrix are entered one after the other. Once the input is taken, print the matrix A and matrix B values.



```
#create two arrays of size given above
arr1 = np.zeros(shape = (matrixSize,matrixSize))
arr2 = np.zeros(shape = (matrixSize,matrixSize))

#take matrix A and matrix B as input for matrix multiplication
for i in range(0,matrixSize):
    for j in range(0,matrixSize):
        arr1[i][j] = input("Enter numbers in the first matrix A, one after the other: ")

for i in range(0,matrixSize):
    for j in range(0,matrixSize):
        arr2[i][j] = input("Enter numbers in the second matrix B, one after the other: ")

print("Below is the matrix_A",arr1)
print("Below is the matrix_B",arr2)
```

3. Now, I have passed matrix A and matrix B as parameters to the "strassen" function.



```
matrix_A = np.array(arr1)
matrix_B = np.array(arr2)

print('Strassens Matrix multiplication result for above matrix_A and matrix_B is : ')
print(strassen(matrix_A, matrix_B))
```

4. The function "strassen" is a recursive function that divides matrix A and matrix B into sub-matrices and computes as per Strassen's sum and product formulas to produce a result matrix C.
<u>Strassen's Algorithm:</u>
i. Firstly, the function takes matrix A and matrix B as input.

CSCE 5150 – Analysis of Computer Algorithms

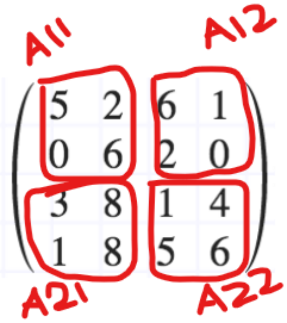
Programming Assignment 1 – Divide and Conquer

Neha Goud Baddam
11519516

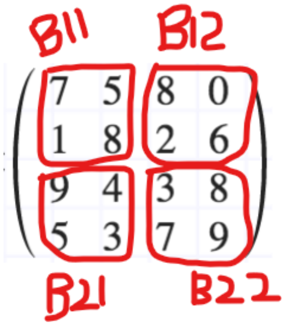
ii. We check if the size of matrix A or matrix B is one, if it is a 1x1 matrix, we do not have to divide it further. Hence, we return a simple multiplied value of matrix A and matrix B.

iii. Now, we check if the size of the matrix is even or odd. If it is odd, we add another column to the matrices A and B for easy division of the matrix to for sub-matrices. If the size is even, we do not modify the matrices.

iv. Now, we divide the matrix into sub-matrices of equal size. Say, if it is a 4x4 matrix, we divide it into 4 sub-matrices. A11 will take the values of the first quadrant of matrix A, A12 will take the second quadrant values of matrix A, A21 will take the third quadrant values of matrix A and A22 will take the fourth quadrant values of matrix A. Similarly, B11, B12, B21, and B22 will take the values of matrix B values of the first, second, third, and fourth quadrants, respectively.

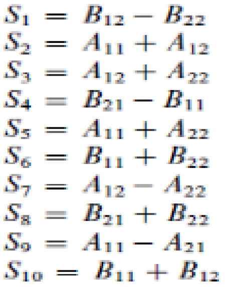


$$\begin{pmatrix} 5 & 2 & 6 & 1 \\ 0 & 6 & 2 & 0 \\ 3 & 8 & 1 & 4 \\ 1 & 8 & 5 & 6 \end{pmatrix}$$



$$\begin{pmatrix} 7 & 5 & 8 & 0 \\ 1 & 8 & 2 & 6 \\ 9 & 4 & 3 & 8 \\ 5 & 3 & 7 & 9 \end{pmatrix}$$

v. Once the matrices are partitioned into sub-matrices, we now calculate the sum values as per Strassen's algorithm.



$$\begin{aligned} S_1 &= B_{12} - B_{22} \\ S_2 &= A_{11} + A_{12} \\ S_3 &= A_{12} + A_{22} \\ S_4 &= B_{21} - B_{11} \\ S_5 &= A_{11} + A_{22} \\ S_6 &= B_{11} + B_{22} \\ S_7 &= A_{12} - A_{22} \\ S_8 &= B_{21} + B_{22} \\ S_9 &= A_{11} - A_{21} \\ S_{10} &= B_{11} + B_{12} \end{aligned}$$

vi. Now, we compute the product values by recursively calling the defined function by passing the sub-matrices and above-computed sum values of matrices as shown below.

CSCE 5150 – Analysis of Computer Algorithms

Programming Assignment 1 – Divide and Conquer

Neha Goud Baddam
11519516

$$\begin{aligned}
P_1 &= \text{STRASSEN}(A_{11}, S_1) \\
P_2 &= \text{STRASSEN}(S_2, B_{22}) \\
P_3 &= \text{STRASSEN}(S_3, B_{11}) \\
P_4 &= \text{STRASSEN}(A_{22}, S_4) \\
P_5 &= \text{STRASSEN}(S_5, S_6) \\
P_6 &= \text{STRASSEN}(S_7, S_8) \\
P_7 &= \text{STRASSEN}(S_9, S_{10})
\end{aligned}$$

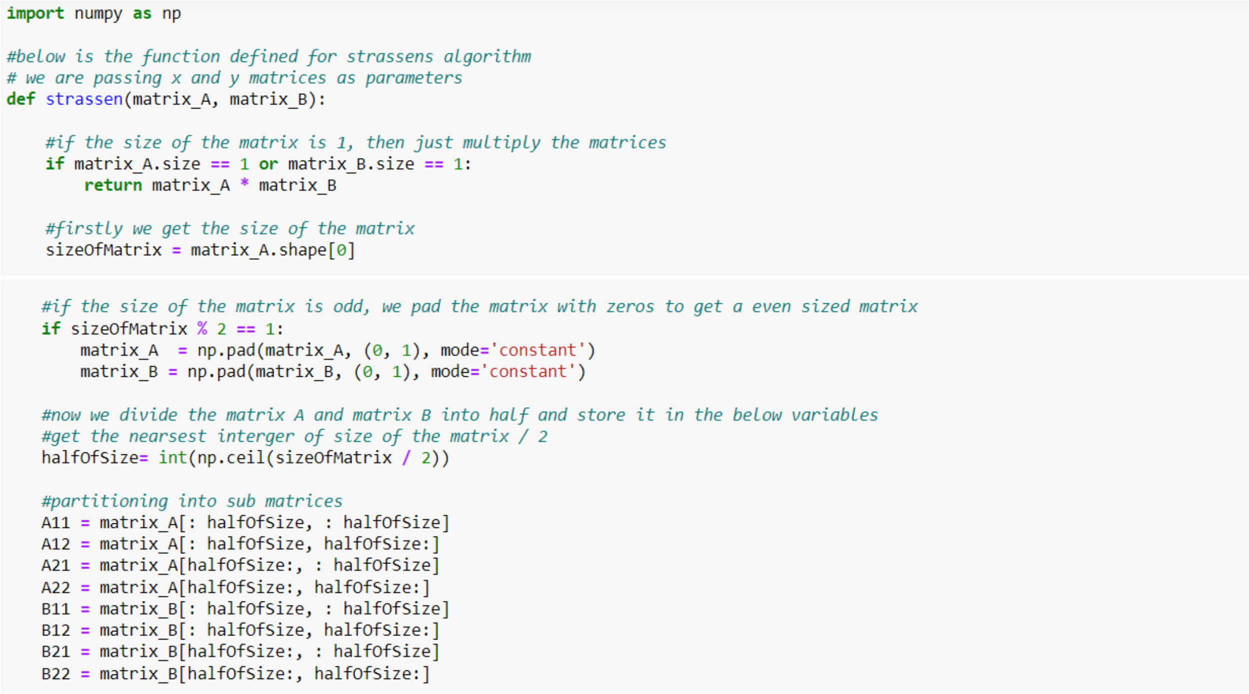
vii. The Strassen function is called recursively until the size of the sub-divided matrix is 1.

viii. Once, the product values are all computed. The resulting matrix is calculated using the following formula.

$$\begin{aligned}
C_{11} &= P_5 + P_4 - P_2 + P_6 \\
C_{12} &= P_1 + P_2 \\
C_{21} &= P_3 + P_4 \\
C_{22} &= P_5 + P_1 - P_3 - P_7
\end{aligned}$$

ix. Now the combined matrix C is the final result of the matrix multiplication of matrix A and matrix B.

Code for Strassen’s matrix multiplication:



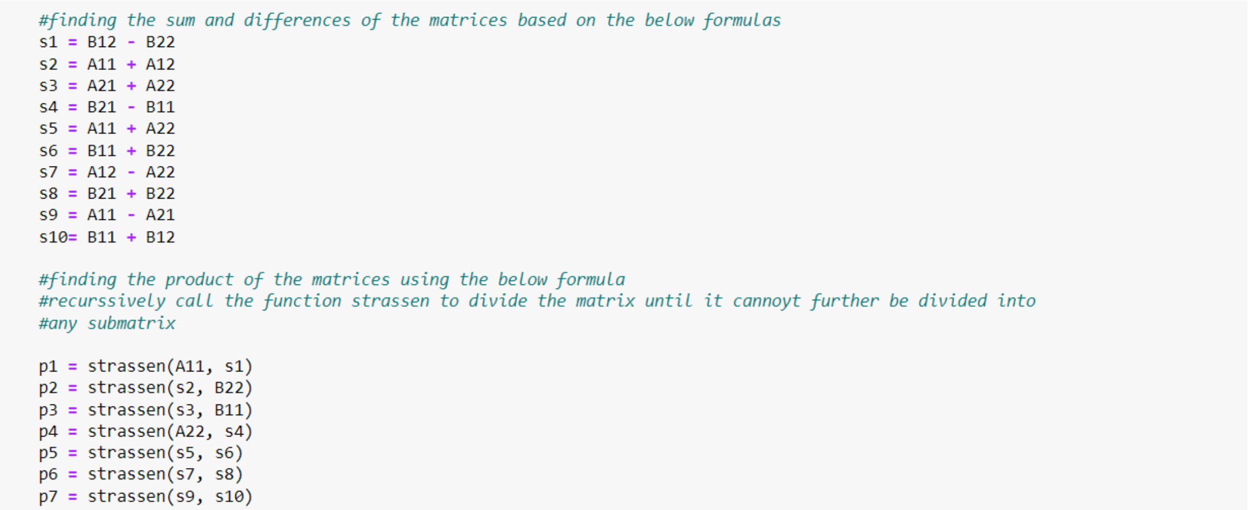
```python
import numpy as np

#below is the function defined for strassens algorithm
# we are passing x and y matrices as parameters
def strassen(matrix_A, matrix_B):

    #if the size of the matrix is 1, then just multiply the matrices
    if matrix_A.size == 1 or matrix_B.size == 1:
        return matrix_A * matrix_B

    #firstly we get the size of the matrix
    sizeOfMatrix = matrix_A.shape[0]
```

```python
    #if the size of the matrix is odd, we pad the matrix with zeros to get a even sized matrix
    if sizeOfMatrix % 2 == 1:
        matrix_A  = np.pad(matrix_A, (0, 1), mode='constant')
        matrix_B = np.pad(matrix_B, (0, 1), mode='constant')

    #now we divide the matrix A and matrix B into half and store it in the below variables
    #get the nearsest interger of size of the matrix / 2
    halfOfSize= int(np.ceil(sizeOfMatrix / 2))

    #partitioning into sub matrices
    A11 = matrix_A[: halfOfSize, : halfOfSize]
    A12 = matrix_A[: halfOfSize, halfOfSize:]
    A21 = matrix_A[halfOfSize:, : halfOfSize]
    A22 = matrix_A[halfOfSize:, halfOfSize:]
    B11 = matrix_B[: halfOfSize, : halfOfSize]
    B12 = matrix_B[: halfOfSize, halfOfSize:]
    B21 = matrix_B[halfOfSize:, : halfOfSize]
    B22 = matrix_B[halfOfSize:, halfOfSize:]
```

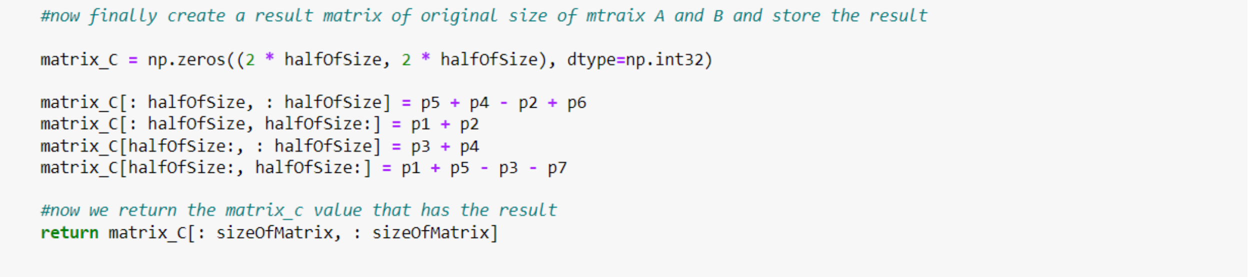
CSCE 5150 – Analysis of Computer Algorithms

Programming Assignment 1 – Divide and Conquer

Neha Goud Baddam
11519516



```
#finding the sum and differences of the matrices based on the below formulas
s1 = B12 - B22
s2 = A11 + A12
s3 = A21 + A22
s4 = B21 - B11
s5 = A11 + A22
s6 = B11 + B22
s7 = A12 - A22
s8 = B21 + B22
s9 = A11 - A21
s10= B11 + B12

#finding the product of the matrices using the below formula
#recurssively call the function strassen to divide the matrix until it cannoyt further be divided into
#any submatrix

p1 = strassen(A11, s1)
p2 = strassen(s2, B22)
p3 = strassen(s3, B11)
p4 = strassen(A22, s4)
p5 = strassen(s5, s6)
p6 = strassen(s7, s8)
p7 = strassen(s9, s10)
```



```
#now finally create a result matrix of original size of mtraix A and B and store the result

matrix_C = np.zeros((2 * halfOfSize, 2 * halfOfSize), dtype=np.int32)

matrix_C[: halfOfSize, : halfOfSize] = p5 + p4 - p2 + p6
matrix_C[: halfOfSize, halfOfSize:] = p1 + p2
matrix_C[halfOfSize:, : halfOfSize] = p3 + p4
matrix_C[halfOfSize:, halfOfSize:] = p1 + p5 - p3 - p7

#now we return the matrix_c value that has the result
return matrix_C[: sizeOfMatrix, : sizeOfMatrix]
```

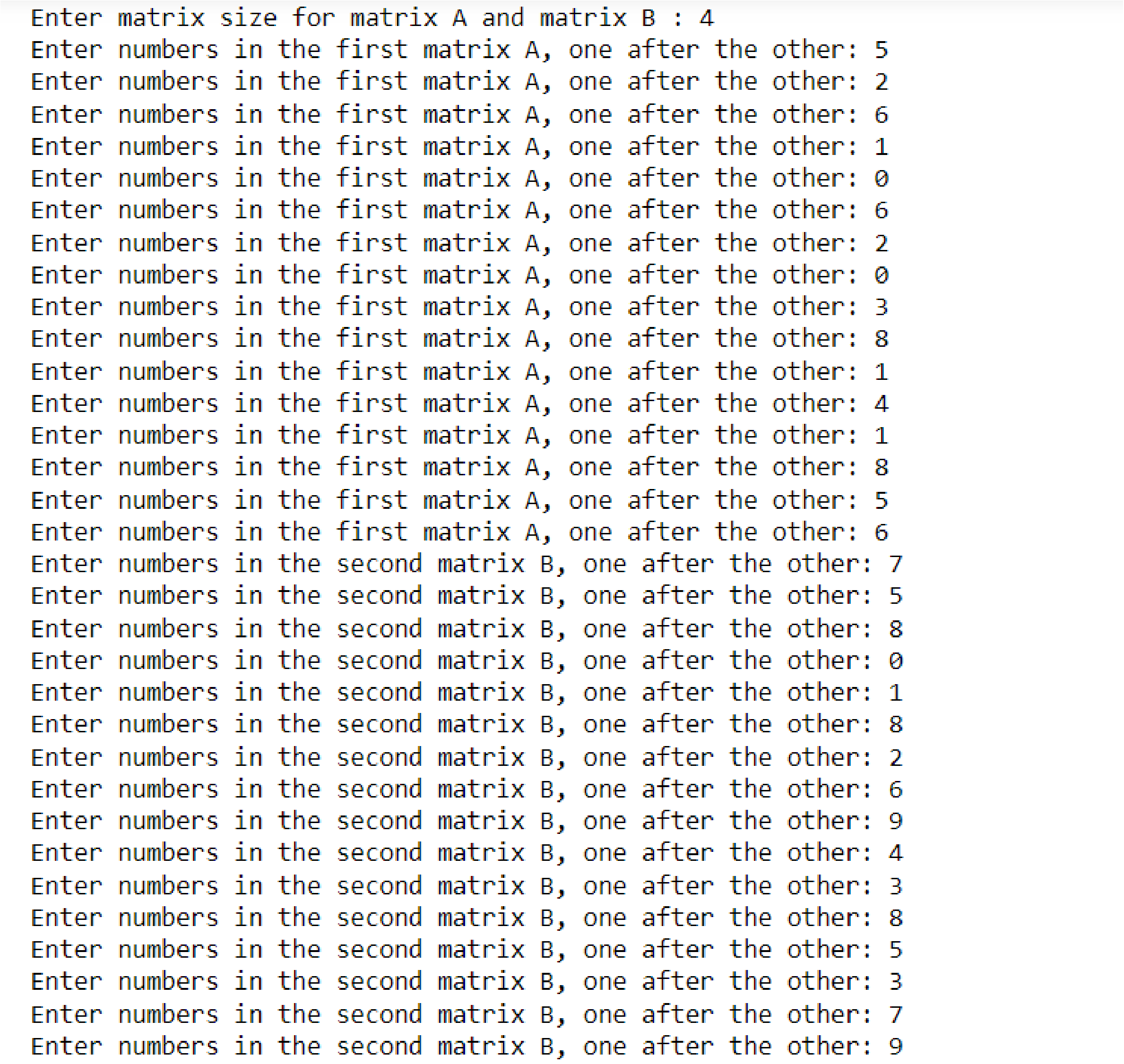
<u>Execution/ Output:</u>

1. **<u>For 4x4 matrix</u>**

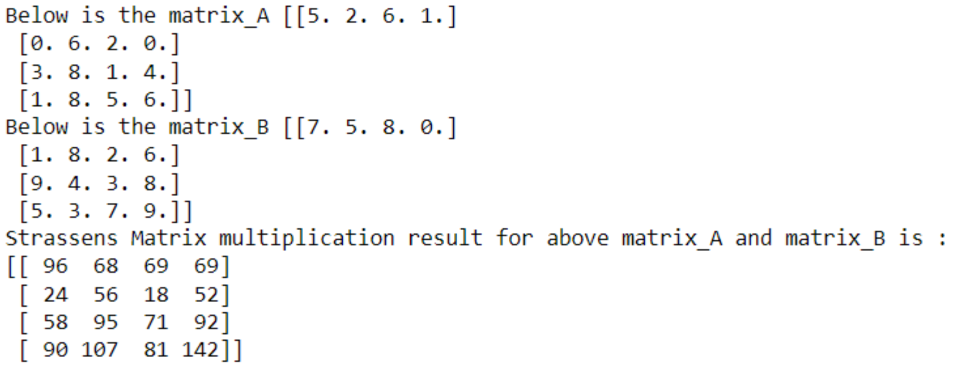
CSCE 5150 – Analysis of Computer Algorithms

Programming Assignment 1 – Divide and Conquer

Neha Goud Baddam
11519516



```
Enter matrix size for matrix A and matrix B : 4
Enter numbers in the first matrix A, one after the other: 5
Enter numbers in the first matrix A, one after the other: 2
Enter numbers in the first matrix A, one after the other: 6
Enter numbers in the first matrix A, one after the other: 1
Enter numbers in the first matrix A, one after the other: 0
Enter numbers in the first matrix A, one after the other: 6
Enter numbers in the first matrix A, one after the other: 2
Enter numbers in the first matrix A, one after the other: 0
Enter numbers in the first matrix A, one after the other: 3
Enter numbers in the first matrix A, one after the other: 8
Enter numbers in the first matrix A, one after the other: 1
Enter numbers in the first matrix A, one after the other: 4
Enter numbers in the first matrix A, one after the other: 1
Enter numbers in the first matrix A, one after the other: 8
Enter numbers in the first matrix A, one after the other: 5
Enter numbers in the first matrix A, one after the other: 6
Enter numbers in the second matrix B, one after the other: 7
Enter numbers in the second matrix B, one after the other: 5
Enter numbers in the second matrix B, one after the other: 8
Enter numbers in the second matrix B, one after the other: 0
Enter numbers in the second matrix B, one after the other: 1
Enter numbers in the second matrix B, one after the other: 8
Enter numbers in the second matrix B, one after the other: 2
Enter numbers in the second matrix B, one after the other: 6
Enter numbers in the second matrix B, one after the other: 9
Enter numbers in the second matrix B, one after the other: 4
Enter numbers in the second matrix B, one after the other: 3
Enter numbers in the second matrix B, one after the other: 8
Enter numbers in the second matrix B, one after the other: 5
Enter numbers in the second matrix B, one after the other: 3
Enter numbers in the second matrix B, one after the other: 7
Enter numbers in the second matrix B, one after the other: 9
```



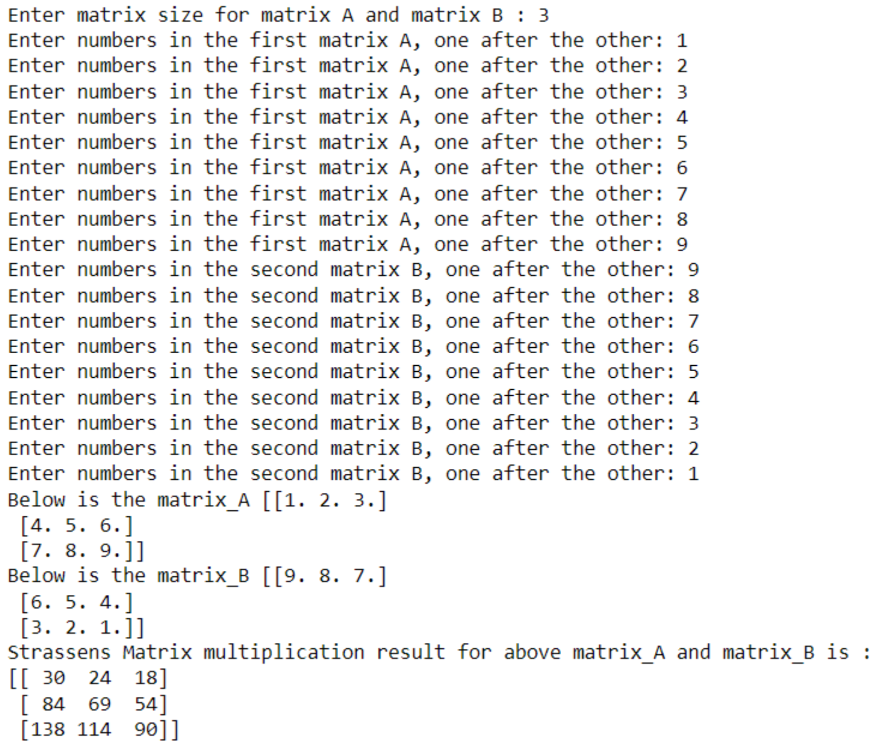
```
Below is the matrix_A [[5. 2. 6. 1.]
 [0. 6. 2. 0.]
 [3. 8. 1. 4.]
 [1. 8. 5. 6.]]
Below is the matrix_B [[7. 5. 8. 0.]
 [1. 8. 2. 6.]
 [9. 4. 3. 8.]
 [5. 3. 7. 9.]]
Strassens Matrix multiplication result for above matrix_A and matrix_B is :
[[ 96  68  69  69]
 [ 24  56  18  52]
 [ 58  95  71  92]
 [ 90 107  81 142]]
```

CSCE 5150 – Analysis of Computer Algorithms

Programming Assignment 1 – Divide and Conquer

Neha Goud Baddam
11519516

2. **For 3x3 matrix:**



```
Enter matrix size for matrix A and matrix B : 3
Enter numbers in the first matrix A, one after the other: 1
Enter numbers in the first matrix A, one after the other: 2
Enter numbers in the first matrix A, one after the other: 3
Enter numbers in the first matrix A, one after the other: 4
Enter numbers in the first matrix A, one after the other: 5
Enter numbers in the first matrix A, one after the other: 6
Enter numbers in the first matrix A, one after the other: 7
Enter numbers in the first matrix A, one after the other: 8
Enter numbers in the first matrix A, one after the other: 9
Enter numbers in the second matrix B, one after the other: 9
Enter numbers in the second matrix B, one after the other: 8
Enter numbers in the second matrix B, one after the other: 7
Enter numbers in the second matrix B, one after the other: 6
Enter numbers in the second matrix B, one after the other: 5
Enter numbers in the second matrix B, one after the other: 4
Enter numbers in the second matrix B, one after the other: 3
Enter numbers in the second matrix B, one after the other: 2
Enter numbers in the second matrix B, one after the other: 1
Below is the matrix_A [[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
Below is the matrix_B [[9. 8. 7.]
 [6. 5. 4.]
 [3. 2. 1.]]
Strassens Matrix multiplication result for above matrix_A and matrix_B is :
[[ 30  24  18]
 [ 84  69  54]
 [138 114  90]]
```