Name	Loukik Tayshete
UID no.	2021700065
Experiment No.	3

AIM:	To implement and compare the Normal and Strassen’s matrix multiplication
Program 1	
PROBLEM STATEMENT :	Normal Matrix Multiplication

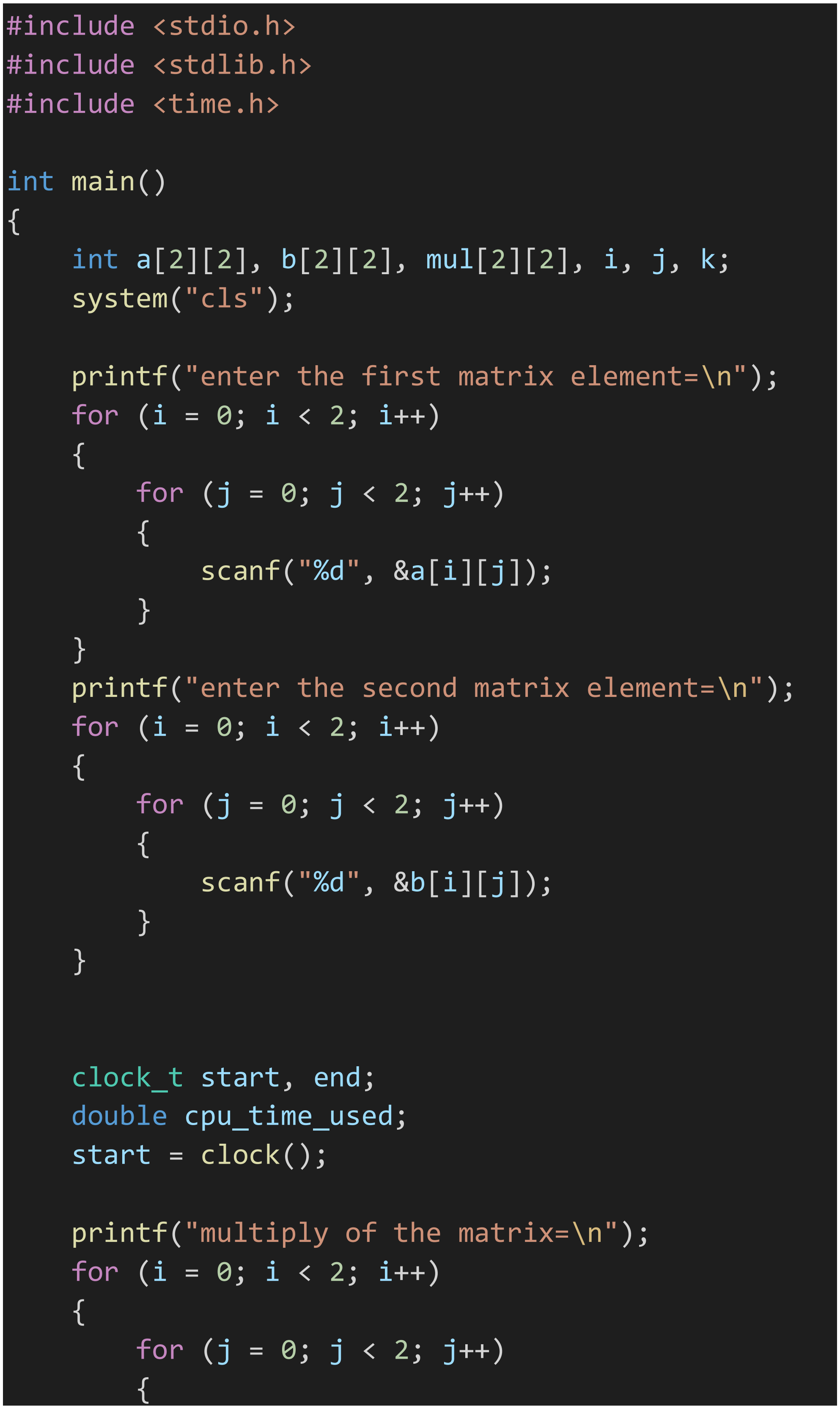
ALGORITHM/ THEORY:

Matrix multiplication in C: We can add, subtract, multiply and divide 2 matrices. To do so, we are taking input from the user for row number, column number, first matrix elements and second matrix elements. Then we are performing multiplication on the matrices entered by the user.

```
void multiply(int A[][N], int B[][N], int C[][N])
{
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            C[i][j] = 0;
            for (int k = 0; k < N; k++)
            {
                C[i][j] += A[i][k]*B[k][j];
            }
        }
    }
}
```

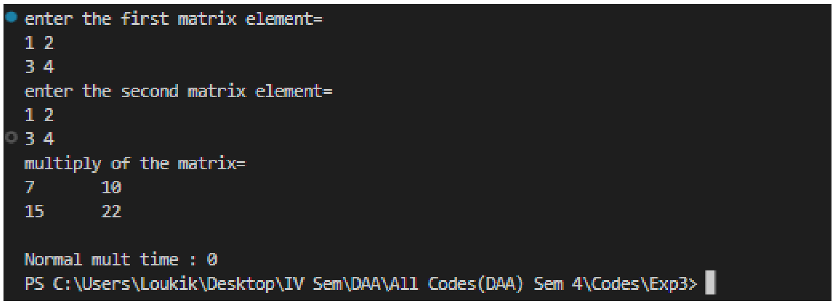
Time Complexity is : O(n^3)

PROGRAM:



```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    int a[2][2], b[2][2], mul[2][2], i, j, k;
    system("cls");

    printf("enter the first matrix element=\n");
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 2; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }
    printf("enter the second matrix element=\n");
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 2; j++)
        {
            scanf("%d", &b[i][j]);
        }
    }


    clock_t start, end;
    double cpu_time_used;
    start = clock();

    printf("multiply of the matrix=\n");
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 2; j++)
        {
            mul[i][j] = 0;
            for (k = 0; k < 2; k++)
            {
                mul[i][j] += a[i][k] * b[k][j];
            }
        }
    }
    // for printing result

    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 2; j++)
        {
            printf("%d\t", mul[i][j]);
        }
        printf("\n");
    }
    end = clock();
    cpu_time_used = ((double)(end - start)) /
CLOCKS_PER_SEC;
    printf("\nNormal mult time : %d\n",
cpu_time_used);

    return 0;
}
```

RESULT:



```
enter the first matrix element=
1 2
3 4
enter the second matrix element=
1 2
3 4
multiply of the matrix=
7       10
15      22

Normal mult time : 0
PS C:\Users\Loukik\Desktop\IV Sem\DAA\All Codes(DAA) Sem 4\Codes\Exp3>
```

Program 2

PROBLEM STATEMENT :

Strassen’s Matrix Multiplication

ALGORITHM/ THEORY:

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.

1. Given two matrices A and B, divide them into four sub-matrices each of size n/2, where n is the size of the original matrices.
2. Compute seven products recursively using these sub-matrices:

M1 = (A11 + A22) x (B11 + B22)
M2 = (A21 + A22) x B11
M3 = A11 x (B12 - B22)
M4 = A22 x (B21 - B11)
M5 = (A11 + A12) x B22
M6 = (A21 - A11) x (B11 + B12)
M7 = (A12 - A22) x (B21 + B22)

3. Compute the four sub-matrices of the result matrix C using these products:

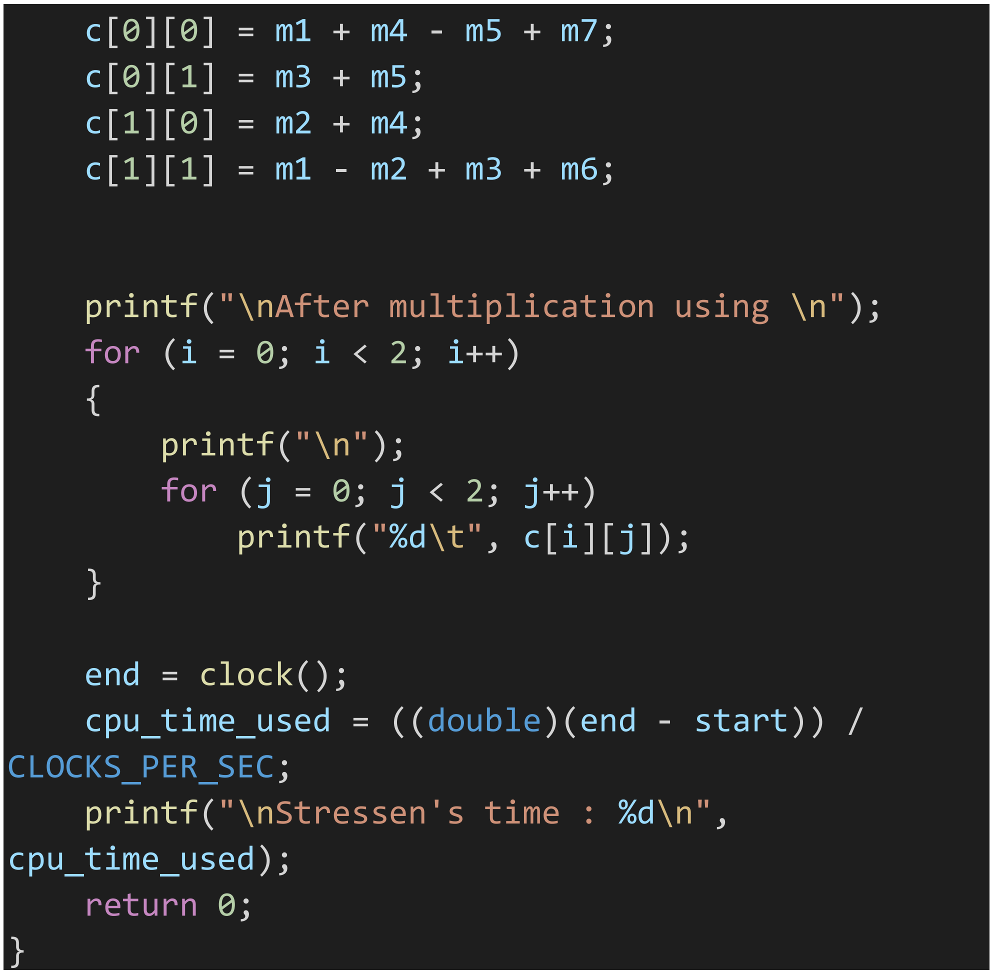
C11 = M1 + M4 - M5 + M7
C12 = M3 + M5
C21 = M2 + M4
C22 = M1 - M2 + M3 + M6

4. Combine these sub-matrices to form the final result matrix C.

PROGRAM:

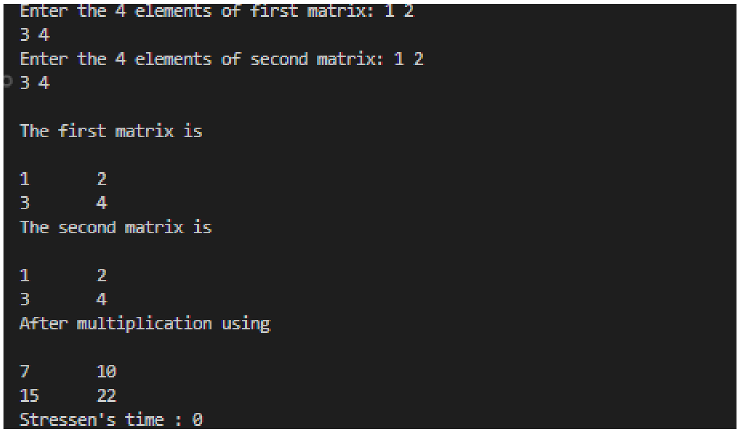
```
#include <stdio.h>

#include <time.h>
int main()
{
    int a[100][100], b[100][100], c[100][100], i,
j;
    int m1, m2, m3, m4, m5, m6, m7;

    printf("Enter the 4 elements of first matrix:
");
    for (i = 0; i < 2; i++)
        for (j = 0; j < 2; j++)
```

```c
            scanf("%d", &a[i][j]);

    printf("Enter the 4 elements of second matrix: 
");
    for (i = 0; i < 2; i++)
        for (j = 0; j < 2; j++)
            scanf("%d", &b[i][j]);

    printf("\nThe first matrix is\n");
    for (i = 0; i < 2; i++)
    {
        printf("\n");
        for (j = 0; j < 2; j++)
            printf("%d\t", a[i][j]);
    }

    printf("\nThe second matrix is\n");
    for (i = 0; i < 2; i++)
    {
        printf("\n");
        for (j = 0; j < 2; j++)
            printf("%d\t", b[i][j]);
    }

    clock_t start, end;
    double cpu_time_used;
    start = clock();

    m1 = (a[0][0] + a[1][1]) * (b[0][0] + b[1][1]);
    m2 = (a[1][0] + a[1][1]) * b[0][0];
    m3 = a[0][0] * (b[0][1] - b[1][1]);
    m4 = a[1][1] * (b[1][0] - b[0][0]);
    m5 = (a[0][0] + a[0][1]) * b[1][1];
    m6 = (a[1][0] - a[0][0]) * (b[0][0] + b[0][1]);
    m7 = (a[0][1] - a[1][1]) * (b[1][0] + b[1][1]);
```



```
    c[0][0] = m1 + m4 - m5 + m7;
    c[0][1] = m3 + m5;
    c[1][0] = m2 + m4;
    c[1][1] = m1 - m2 + m3 + m6;


    printf("\nAfter multiplication using \n");
    for (i = 0; i < 2; i++)
    {
        printf("\n");
        for (j = 0; j < 2; j++)
            printf("%d\t", c[i][j]);
    }

    end = clock();
    cpu_time_used = ((double)(end - start)) /
CLOCKS_PER_SEC;
    printf("\nStressen's time : %d\n",
cpu_time_used);
    return 0;
}
```

RESULT:



```
Enter the 4 elements of first matrix: 1 2
3 4
Enter the 4 elements of second matrix: 1 2
3 4

The first matrix is

1       2
3       4
The second matrix is

1       2
3       4
After multiplication using

7       10
15      22
Stressen's time : 0
```

CONCLUSION:	We can say that the time required for Strassen’s Algo is slight less than that of normal method as the time complexity for strassen’s is O(n^2.807) and for normal it is O(n^3).