Name	Siddhesh Sonar
UID no.	2021700063
Experiment No.	3

AIM:	To implement and compare the Normal and Strassen's matrix multiplication
Program 1	
PROBLEM STATEMENT :	To implement normal matrix multiplication

ALGORITHM/ THEORY:	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)

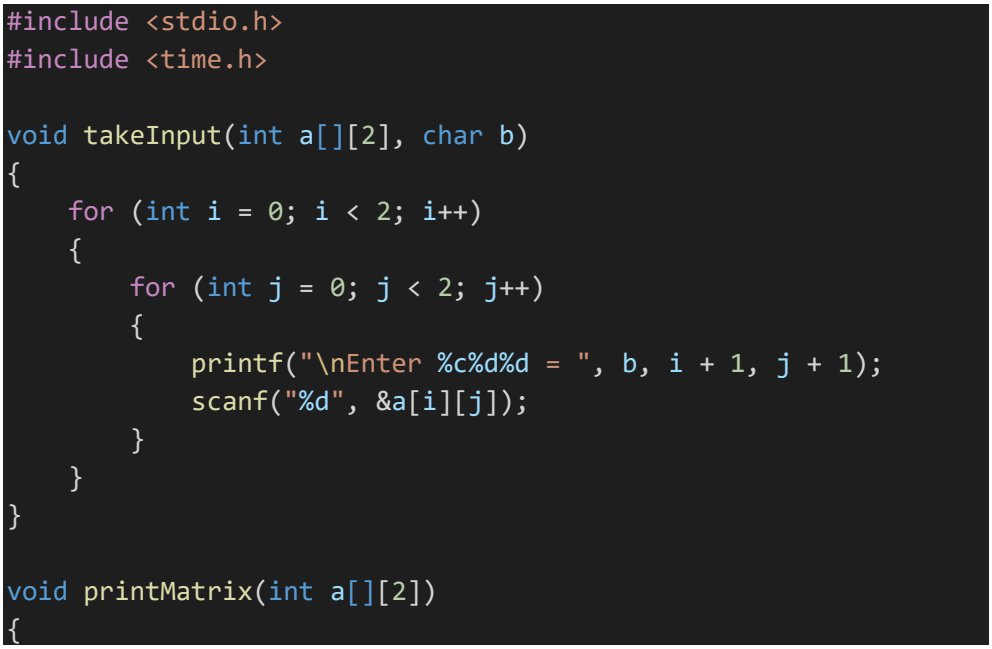
PROGRAM:

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

void takeInput(int a[][2], char b)
{
    for (int i = 0; i < 2; i++)
    {
        for (int j = 0; j < 2; j++)
        {
            printf("\nEnter %c%d%d = ", b, i + 1, j + 1);
            scanf("%d", &a[i][j]);
        }
    }
}

void printMatrix(int a[][2])
{
    for (int i = 0; i < 2; i++)
    {
        printf("\n");
        for (int j = 0; j < 2; j++)
            printf("%d\t", a[i][j]);
    }
}

int main()
{
    int a[2][2], b[2][2], c[2][2];
    int c1, c2, c3, c4, c5, c6, c7;
    clock_t start, end;
    double time_taken;
    printf("\nEnter the 4 elements of first matrix\n");
    takeInput(a, 'a');
    printf("\nEnter the 4 elements of second matrix\n");
    takeInput(b, 'b');
    printf("\n\nThe first matrix is\n");
    printMatrix(a);
    printf("\n\nThe second matrix is\n");
    printMatrix(b);
    start = clock();
    c1 = (a[0][0] + a[1][1]) * (b[0][0] + b[1][1]);
    c2 = (a[1][0] + a[1][1]) * b[0][0];
    c3 = a[0][0] * (b[0][1] - b[1][1]);
    c4 = a[1][1] * (b[1][0] - b[0][0]);
```

```
    c5 = (a[0][0] + a[0][1]) * b[1][1];
    c6 = (a[1][0] - a[0][0]) * (b[0][0] + b[0][1]);
    c7 = (a[0][1] - a[1][1]) * (b[1][0] + b[1][1]);

    c[0][0] = c1 + c4 - c5 + c7;
    c[0][1] = c3 + c5;
    c[1][0] = c2 + c4;
    c[1][1] = c1 - c2 + c3 + c6;

    printf("\n\nAfter performing multiplication\n");
    printMatrix(c);
    end = clock();
    time_taken = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("\nStressen's time : %f\n", time_taken);
    return 0;
}
```

RESULT:

```
Microsoft Windows [Version 10.0.22621.1265]
(c) Microsoft Corporation. All rights reserved.

C:\Siddhesh\DAA>cd "c:\Siddhesh\DAA\DAA_Exp_3\" && gcc normal_matrix.c

Enter the 4 elements of first matrix

Enter a11 = 1

Enter a12 = 2

Enter a21 = 3

Enter a22 = 4

Enter the 4 elements of second matrix

Enter b11 = 5

Enter b12 = 6

Enter b21 = 7

Enter b22 = 8


The first matrix is

1       2
3       4

The second matrix is

5       6
7       8

Matrix after multiplication:

19      22
43      50

Normal mult time : 0.000000

c:\Siddhesh\DAA\DAA_Exp_3>
```

Program 2	
PROBLEM STATEMENT :	To implement 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:	

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

void takeInput(int a[][2], char b)
{
    for (int i = 0; i < 2; i++)
    {
        for (int j = 0; j < 2; j++)
        {
            printf("\nEnter %c%d%d = ", b, i + 1, j + 1);
            scanf("%d", &a[i][j]);
        }
    }
}

void printMatrix(int a[][2])
{
```

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

int main()
{
    int a[2][2], b[2][2], c[2][2];
    int c1, c2, c3, c4, c5, c6, c7;
    clock_t start, end;
    double time_taken;
    printf("\nEnter the 4 elements of first matrix\n");
    takeInput(a, 'a');
    printf("\nEnter the 4 elements of second matrix\n");
    takeInput(b, 'b');
    printf("\n\nThe first matrix is\n");
    printMatrix(a);
    printf("\n\nThe second matrix is\n");
    printMatrix(b);
    start = clock();
    c1 = (a[0][0] + a[1][1]) * (b[0][0] + b[1][1]);
    c2 = (a[1][0] + a[1][1]) * b[0][0];
    c3 = a[0][0] * (b[0][1] - b[1][1]);
    c4 = a[1][1] * (b[1][0] - b[0][0]);
    c5 = (a[0][0] + a[0][1]) * b[1][1];
    c6 = (a[1][0] - a[0][0]) * (b[0][0] + b[0][1]);
    c7 = (a[0][1] - a[1][1]) * (b[1][0] + b[1][1]);

    c[0][0] = c1 + c4 - c5 + c7;
    c[0][1] = c3 + c5;
    c[1][0] = c2 + c4;
    c[1][1] = c1 - c2 + c3 + c6;

    printf("\n\nAfter performing multiplication\n");
    printMatrix(c);
    end = clock();
    time_taken = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("\n\nStressen's Multiplication time : %f\n",
time_taken);
    return 0;
}
```

RESULT:

```
C:\Siddhesh\DAA>cd "c:\Siddhesh\DAA\DAA_Exp_3\" && gcc stress_matrix.c

Enter the 4 elements of first matrix

Enter a11 = 1

Enter a12 = 2

Enter a21 = 3

Enter a22 = 4

Enter the 4 elements of second matrix

Enter b11 = 5

Enter b12 = 6

Enter b21 = 7

Enter b22 = 8


The first matrix is

1       2
3       4

The second matrix is

5       6
7       8

After performing multiplication

19      22
43      50

Stressen's Multiplication time : 0.000000

c:\Siddhesh\DAA\DAA_Exp_3>
```

CONCLUSION:	Successfully implemented Strassen's matrix multiplication in C program and also found that the time required for Strassen's Algo (time complexity = O(n^2.807)) is slightly less than that required for normal method (time complexity = O(n^3)).