Name	Siddiqui Hawaiza Sabeel
UID no.	2022701010
Experiment No.	3
SUBJECT	DAA

AIM:	**Divide and Conquer – Strassen's Matrix Multiplication**
Program 1	

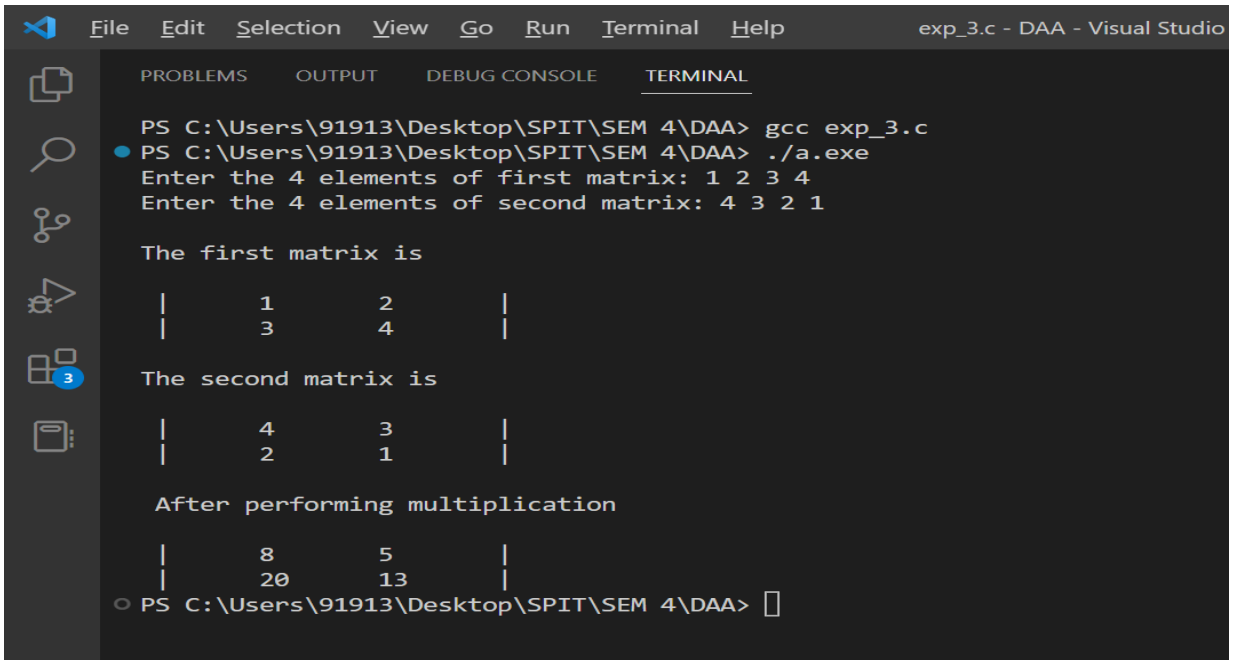
ALGORITHM/ THEORY:	The algorithm for Strassen's matrix Multiplication is as follows:
	
	Algorithm Strass(n, x, y, z)
	
	begin
	If n = threshold then compute
	C = x * y is a conventional matrix.
	Else
	Partition a into four sub matrices a00, a01, a10, a11.
	Partition b into four sub matrices b00, b01, b10, b11.
	Strass (n/2, a00 + a11, b00 + b11, d1)
	Strass (n/2, a10 + a11, b00, d2)
	Strass (n/2, a00, b01 – b11, d3)
	Strass (n/2, a11, b10 – b00, d4)
	Strass (n/2, a00 + a01, b11, d5)
	Strass (n/2, a10 – a00, b00 + b11, d6)
	Strass (n/2, a01 – a11, b10 + b11, d7)
	
	C = d1+d4-d5+d7 d3+d5
	d2+d4 d1+d3-d2-d6
	
	end if
	return (C)
	end.
	
	Using the Master Theorem with **T(n) = 8T(n/2) + O(n^2)** we still get a runtime of **O(n^3)**.

	But Strassen came up with a solution where we don't need 8 recursive calls but can be done in only 7 calls and some extra addition and subtraction operations. Following are the formulae that are to be used for matrix multiplication. 1. D1 = (a11 + a22) * (b11 + b22) 2. D2 = (a21 + a22)*b11 3. D3 = (b12 – b22)*a11 4. D4 = (b21 – b11)*a22 5. D5 = (a11 + a12)*b22 6. D6 = (a21 – a11) * (b11 + b12) 7. D7 = (a12 – a22) * (b21 + b22)

PROGRAM:	`#include <stdio.h>`
	`int main()`
	`{`
	`  int a[2][2], b[2][2], c[2][2], 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++)`
	`  {`
	`  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 | \t");`

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

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

    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("\n\n After performing multiplication \n");
    for (i = 0; i < 2; i++)
    {
        printf("\n | \t");
        for (j = 0; j < 2; j++)
        {
            printf("%d\t", c[i][j]);
        }
        printf("|");
    }
    printf("\n");
    return 0;
}
```

RESULT:



CONCLUSION:	We understood Strassen's Multiplication algorithm and found that Strassen's Multiplication algorithm is better than the standard method of Square Matrix Multiplication. Also, the time complexity of Strassen's Algorithm is lesser than the standard method of Square Matrix Multiplication.