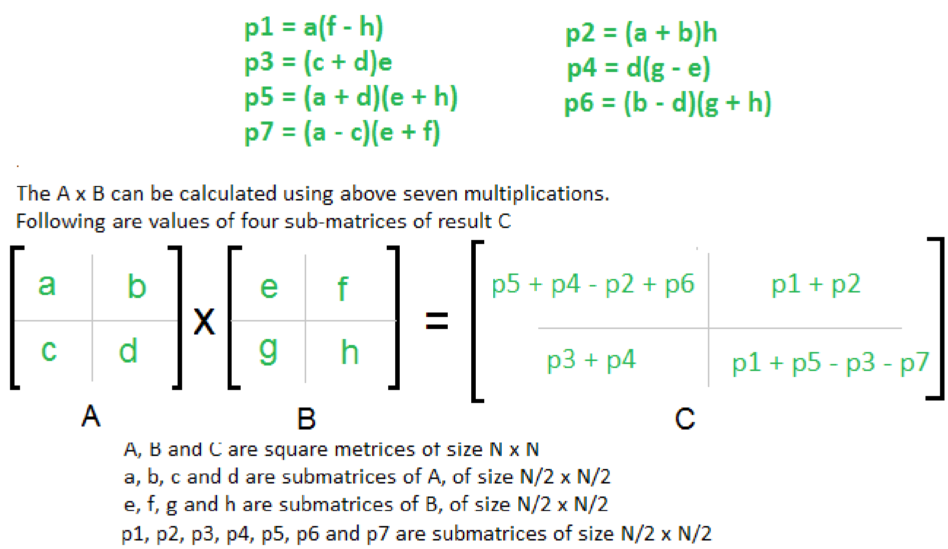
EXPERIMENT 3

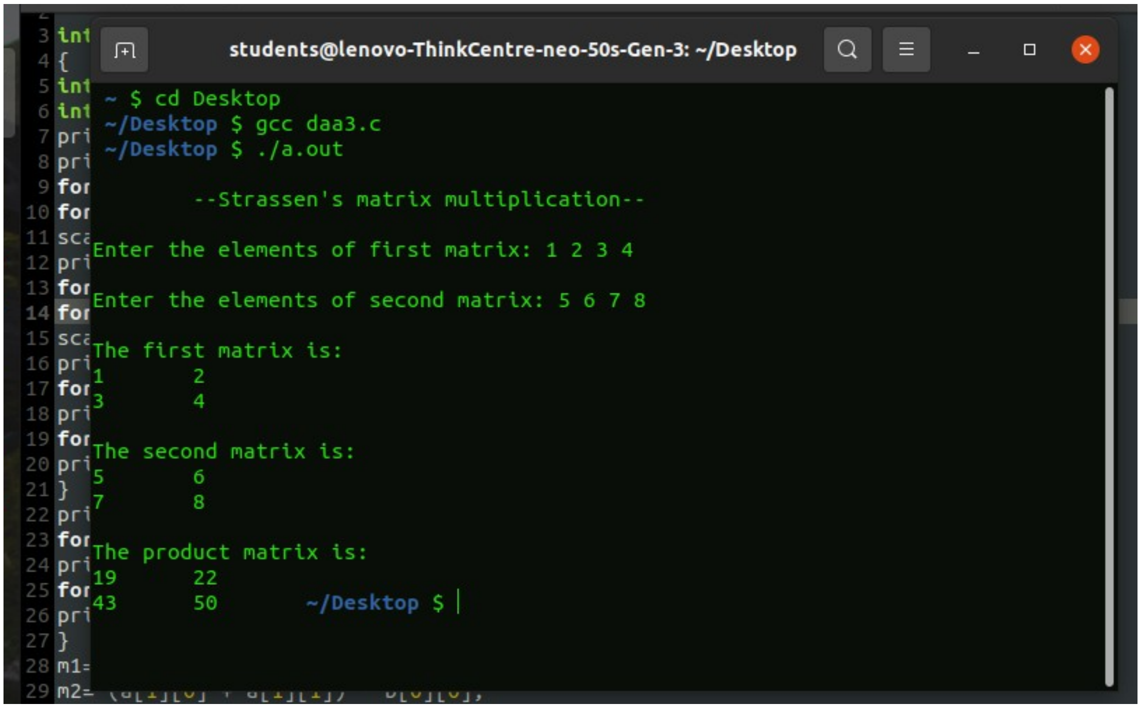
Name	Aditi Nilesh Bhutada
UID no.	2021700009
Experiment No.	3

AIM:	Strassen’s matrix multiplication
PROGRAM	
PROBLEM STATEMENT :	Implement 2*2 matrix multiplication using divide and conquer method-strassen’s matrix multiplication
THEORY:	In the above divide and conquer method, the main component for high time complexity is 8 recursive calls. The idea of Strassen’s method is to reduce the number of recursive calls to 7. Strassen’s method is similar to above simple divide and conquer method in the sense that this method also divide matrices to sub-matrices of size N/2 x N/2 as shown in the above diagram, but in Strassen’s method, the four sub-matrices of result are calculated using following formulae. 

	Time Complexity of Strassen's Method: Addition and Subtraction of two matrices takes $O(N^2)$ time. So time complexity can be written as $T(N) = 7T(N/2) + O(N^2)$ From Master's Theorem, time complexity of above method is $O(N^{Log7})$ which is approximately $O(N^{2.8074})$ Generally Strassen's Method is not preferred for practical applications for following reasons. 1. The constants used in Strassen's method are high and for a typical application Naive method works better. 2. For Sparse matrices, there are better methods especially designed for them. 3. The submatrices in recursion take extra space. 4. Because of the limited precision of computer arithmetic on noninteger values, larger errors accumulate in Strassen's algorithm than in Naive Method
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("\n\t--Strassen's matrix multiplication--\n"); printf("\nEnter the elements of first matrix: "); for(i = 0;i < 2; i++) for(j = 0;j < 2; j++) scanf("%d", &a[i][j]); printf("\nEnter the 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: "); for(i = 0; i < 2; i++){ printf("\n"); for(j = 0; j < 2; j++) printf("%d\t", a[i][j]); } printf("\n\nThe second matrix is: "); for(i = 0;i < 2; i++){ printf("\n"); for(j = 0;j < 2; j++) printf("%d\t", b[i][j]); } 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\nThe product matrix is: "); for(i = 0; i < 2 ; i++){ printf("\n"); for(j = 0;j < 2; j++) printf("%d\t", c[i][j]); } return 0; } ```

RESULT:



CONCLUSION/ ANALYSIS:	In this experiment i understood that the time taken by strassens matrix multiplication is less as it has average time complexity of n^2.81 i.e n^log7 which is less than normal multiplication i.e n^3 Time Complexity of Strassen's Method: Addition and Subtraction of two matrices takes $O(N^2)$ time. So time complexity can be written as $T(N) = 7T(N/2) + O(N^2)$ From Master's Theorem, time complexity of above method is $O(N^{Log7})$ which is approximately $O(N^{2.8074})$