**ASSIGNMENT 6**

Name : Bryce Ferreira Roll no.: 2203129

Enrollment no.: MITU20BTCS0076 Division: CORE 3

**Multiplication of two matrices using threading:**

#include <bits/stdc++.h>

#include <ctime>

using namespace std;

// maximum size of matrix

#define MAX 4

// maximum number of threads

#define MAX\_THREAD 4

int matA[MAX][MAX];

int matB[MAX][MAX];

int matC[MAX][MAX];

int step\_i = 0;

void\* multi(void\* arg)

{

    int i = step\_i++; //i denotes row number of resultant matC

    for (int j = 0; j < MAX; j++)

      for (int k = 0; k < MAX; k++)

        matC[i][j] += matA[i][k] \* matB[k][j];

}

// Driver Code

int main()

{

    // Generating random values in matA and matB

    for (int i = 0; i < MAX; i++) {

        for (int j = 0; j < MAX; j++) {

            matA[i][j] = rand() % 10;

            matB[i][j] = rand() % 10;

        }

    }

    // Displaying matA

    cout << endl

         << "Matrix A" << endl;

    for (int i = 0; i < MAX; i++) {

        for (int j = 0; j < MAX; j++)

            cout << matA[i][j] << " ";

        cout << endl;

    }

    // Displaying matB

    cout << endl

         << "Matrix B" << endl;

    for (int i = 0; i < MAX; i++) {

        for (int j = 0; j < MAX; j++)

            cout << matB[i][j] << " ";

        cout << endl;

    }

    // declaring four threads

    pthread\_t threads[MAX\_THREAD];

    // Creating four threads, each evaluating its own part

    for (int i = 0; i < MAX\_THREAD; i++) {

        int\* p;

        pthread\_create(&threads[i], NULL, multi, (void\*)(p));

    }

    // joining and waiting for all threads to complete

    for (int i = 0; i < MAX\_THREAD; i++)

        pthread\_join(threads[i], NULL);

    // Displaying the result matrix

    cout << endl

         << "Multiplication of A and B" << endl;

    for (int i = 0; i < MAX; i++) {

        for (int j = 0; j < MAX; j++)

            cout << matC[i][j] << " ";

        cout << endl;

    }

cout<<”Time taken: “;

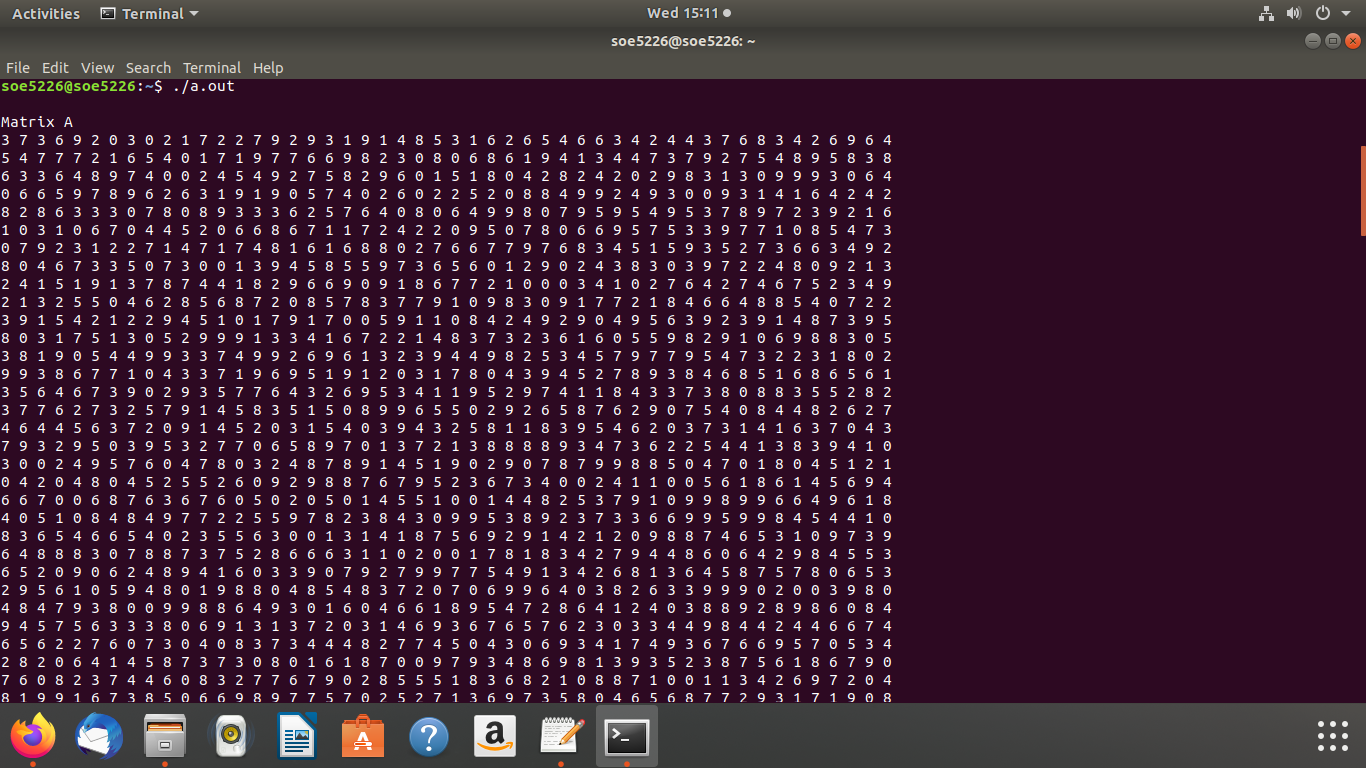
cout<<time(NULL);

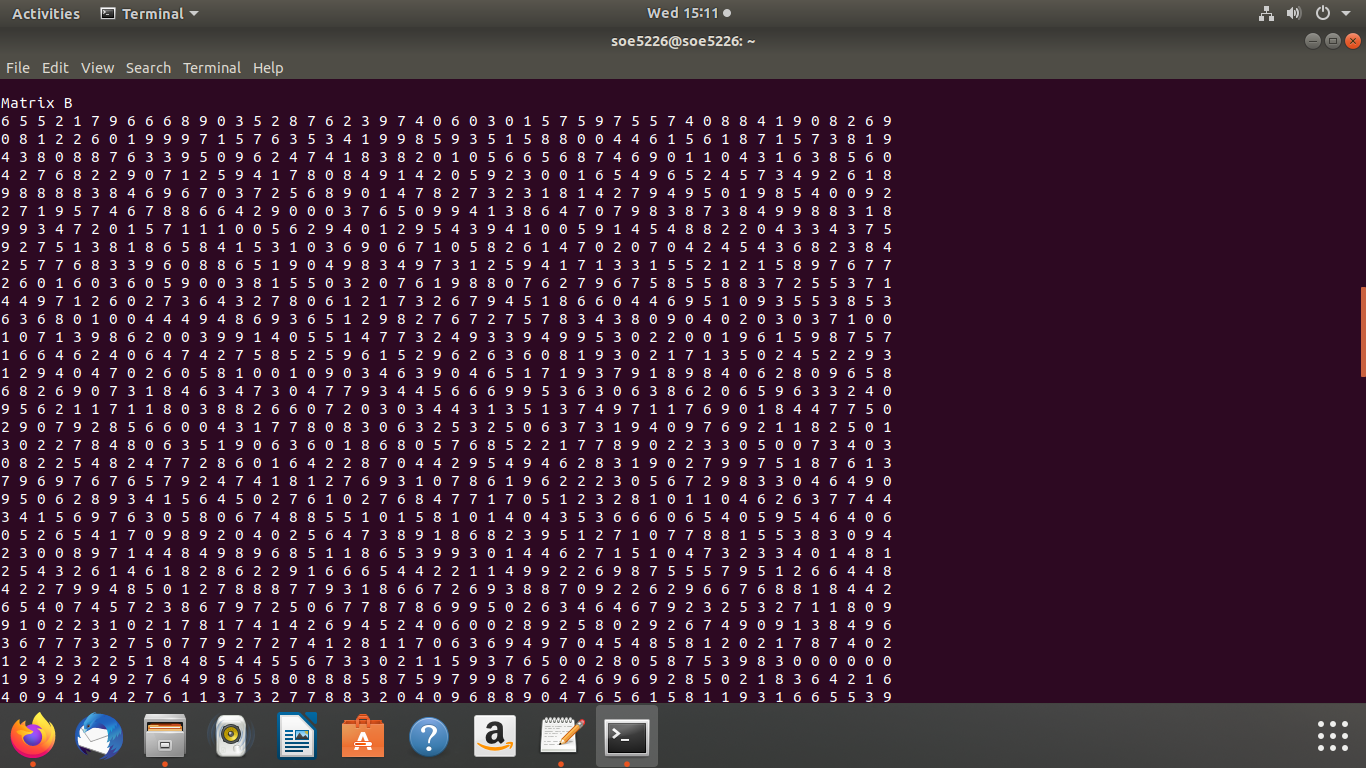
    return 0;

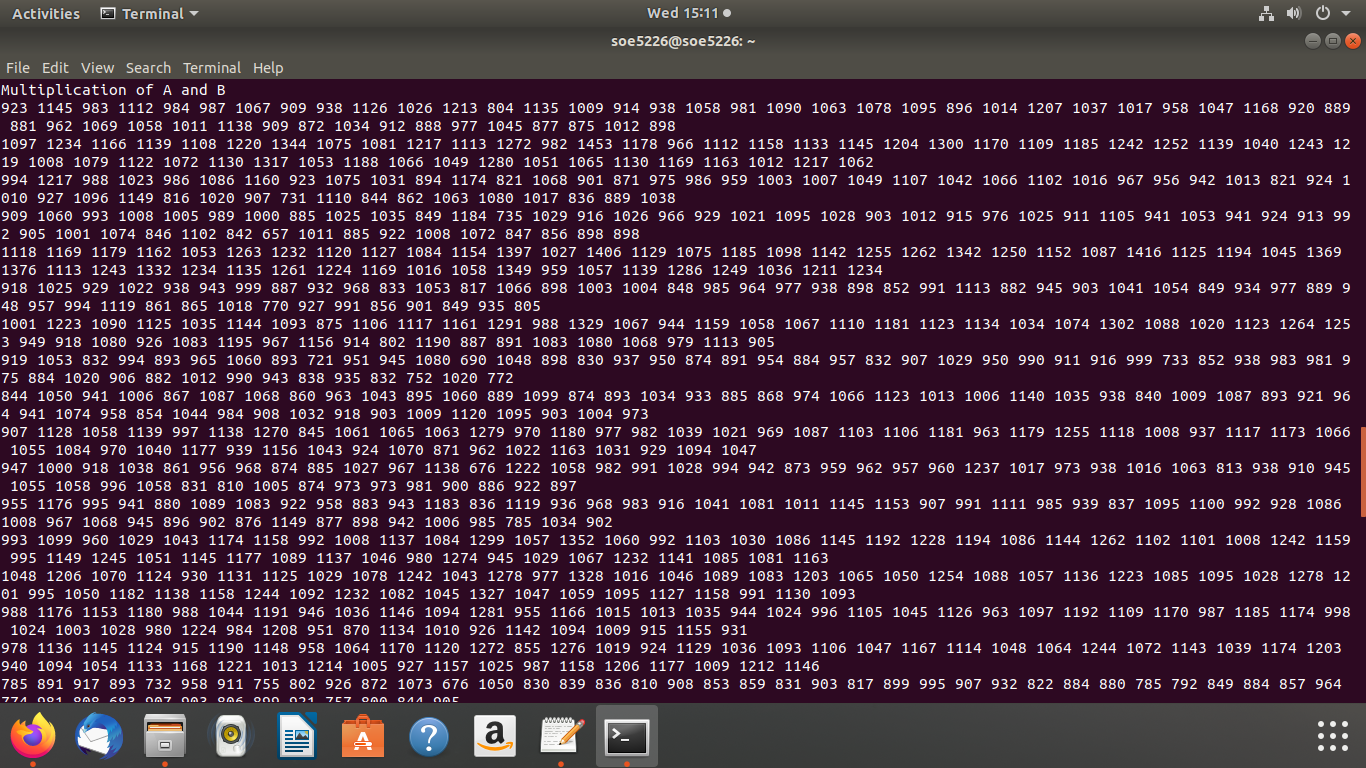
}

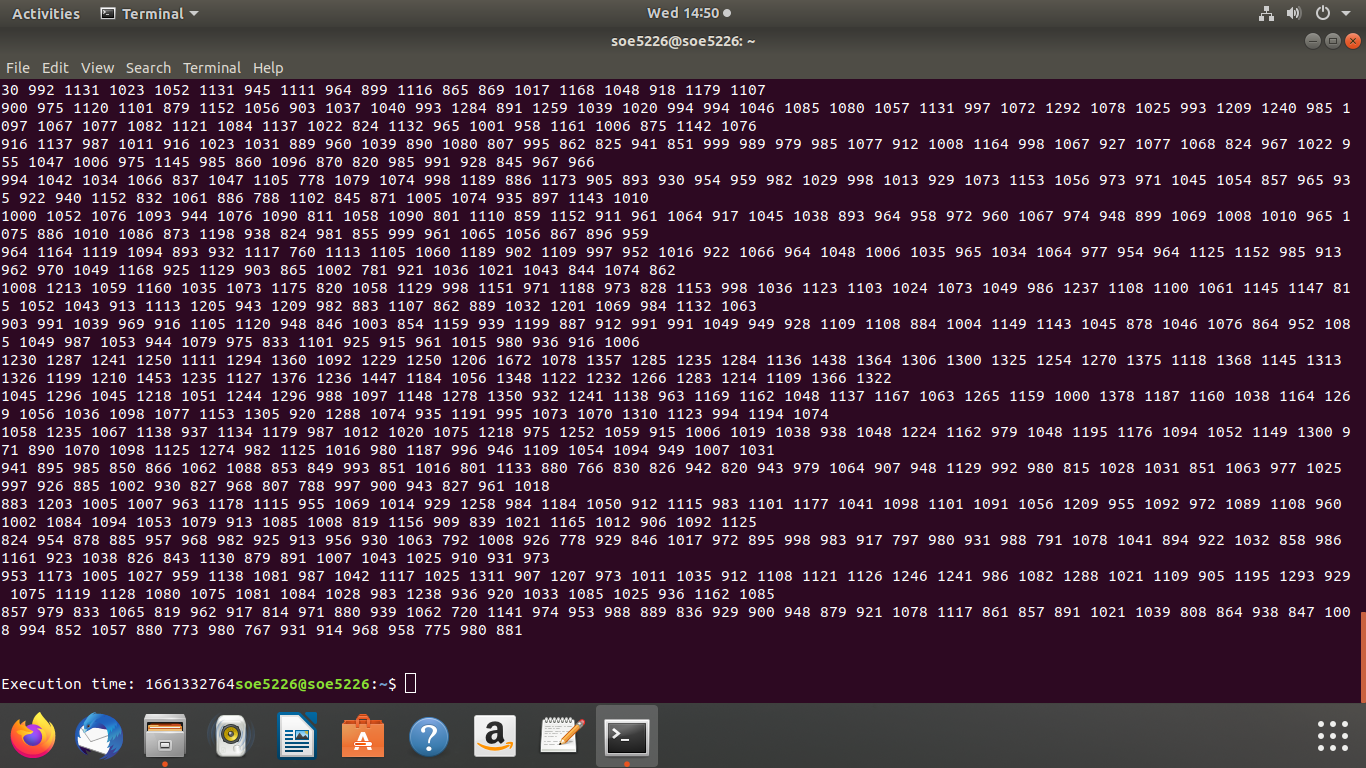
ss1.png

ss2.png









**Normal Matrix Multiplication:**

#include <iostream>

#include <ctime>

using namespace std;

int main()

{

int a[10][10], b[10][10], mult[10][10], r1, c1, r2, c2, i, j, k;

//cout<< "Enter rows and columns for first matrix: ";

//cin>> r1 >>c1;

r1 = 4;

c1 = 4;

//cout<< "Enter rows and columns for second matrix: ";

//cin>> r2 >>c2;

r2 = 4;

c2 = 4;

// If column of first matrix in not equal to row of second matrix,

// ask the user to enter the size of matrix again.

//while (c1!=r2)

// {

//cout<< "Error! column of first matrix not equal to row of second.";

// cout<< "Enter rows and columns for first matrix: ";

//cin>> r1 >>c1;

// cout<< "Enter rows and columns for second matrix: ";

// cin>> r2 >>c2;

// }

// Storing elements of first matrix.

for(i = 0; i< r1; ++i)

for(j = 0; j < c1; ++j)

{

//cout<< "Enter element a" <<i + 1 << j + 1 <<" : ";

//cin>> a[i][j];

a[i][j] = rand() % 10;

}

cout<<endl<< "elements of matrix 1:" <<endl;

for(i = 0; i< r1; ++i){

for(j = 0; j < c1; ++j)

{

//cout<< "Enter element a" <<i + 1 << j + 1 <<" : ";

//cin>> a[i][j];

cout<<" "<< a[i][j];

}

cout<<endl;

};

cout<<endl;

// Storing elements of second matrix.

for(i = 0; i< r2; ++i)

for(j = 0; j < c2; ++j)

{

//cout<< "Enter element b" <<i + 1 << j + 1 <<" : ";

//cin>> b[i][j]

b[i][j] = rand() % 10;

}

cout<<endl<< " elements of matrix 2:" <<endl;

for(i = 0; i< r2; ++i){

for(j = 0; j < c2; ++j)

{

//cout<< "Enter element b" <<i + 1 << j + 1 <<" : ";

//cin>> b[i][j]

cout<< " "<< b[i][j] ;

}

cout<<endl;

};

cout<<endl;

// Initializing elements of matrix mult to 0.

for(i = 0; i< r1; ++i)

for(j = 0; j < c2; ++j)

{

mult[i][j]=0;

}

// Multiplying matrix a and b and storing in array mult.

for(i = 0; i< r1; ++i)

for(j = 0; j < c2; ++j)

for(k = 0; k < c1; ++k)

{

mult[i][j] += a[i][k] \* b[k][j];

}

// Displaying the multiplication of two matrix.

cout<<endl<< "Output Matrix: " <<endl;

for(i = 0; i< r1; ++i)

for(j = 0; j < c2; ++j)

{

cout<< " " <<mult[i][j];

if(j == c2-1)

cout<<endl;

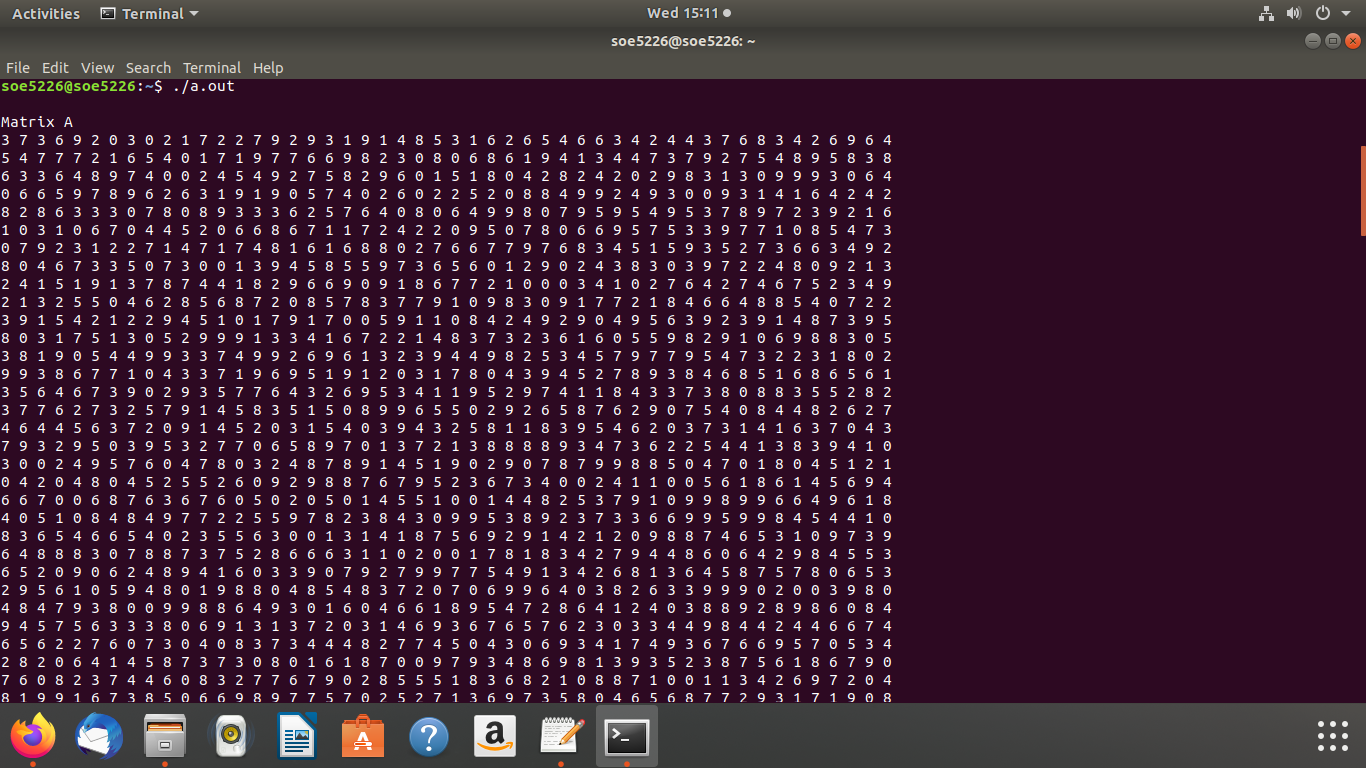
}

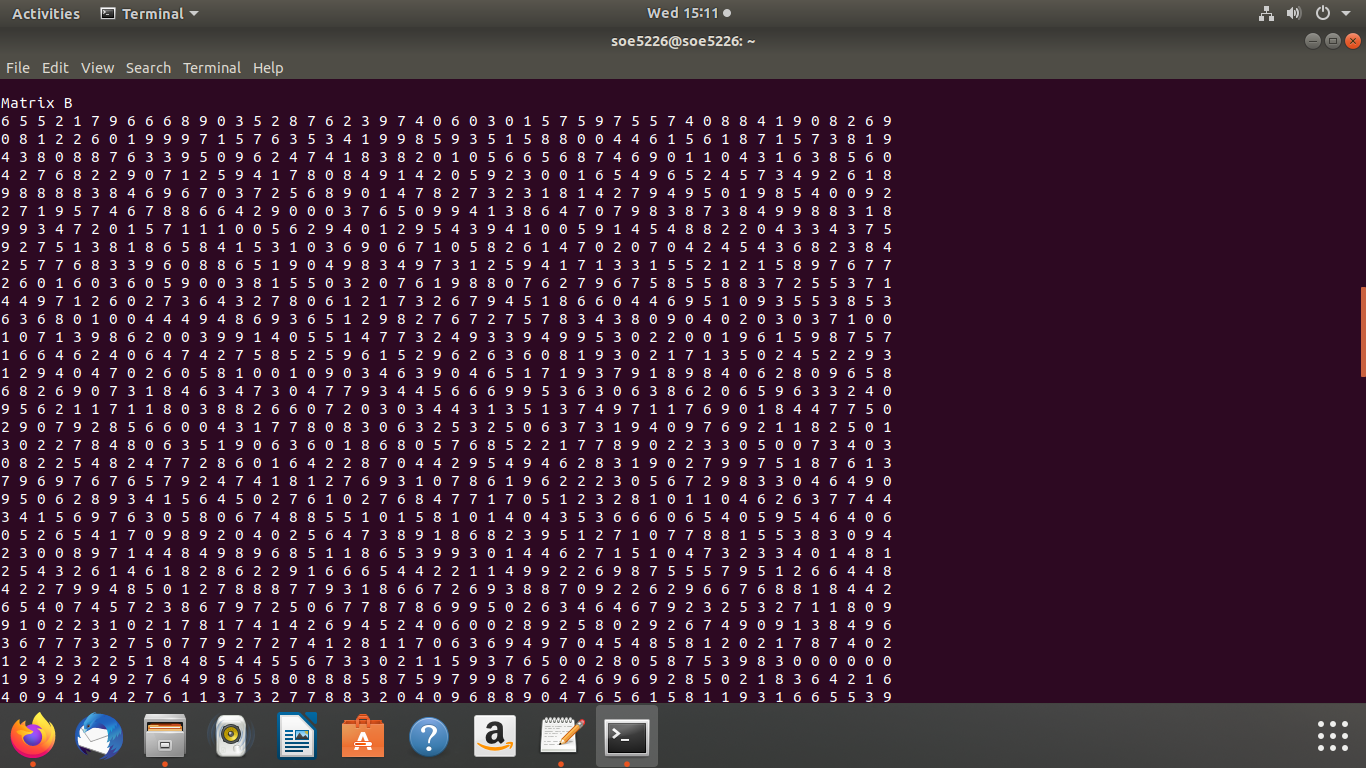
cout<<"time is :";

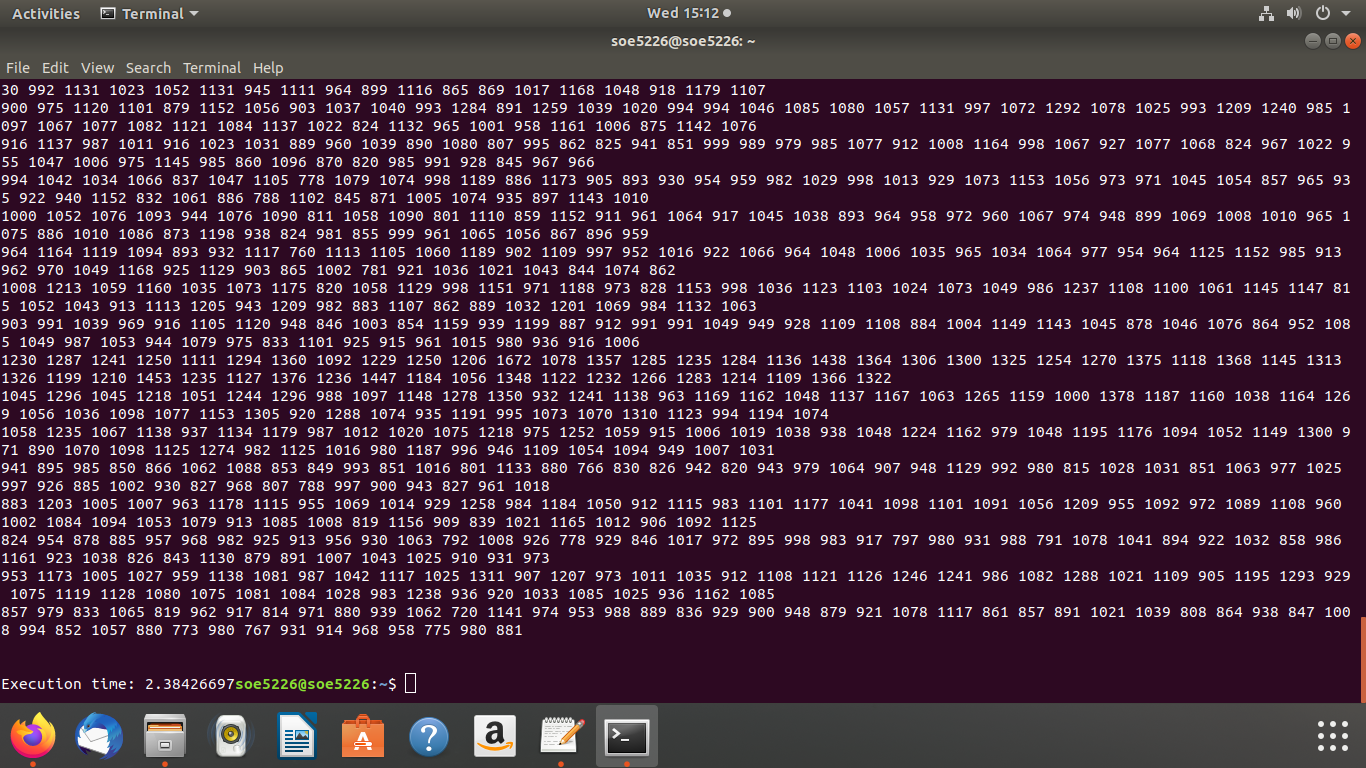
cout<<time(NULL);

return 0;

}







**CONCLUSION :**

we have successfully implemented matrix multiplication using pthread and normal method and calculated the time complexity for both the methods.