# CSE4001 - Parallel and Distributed Computing

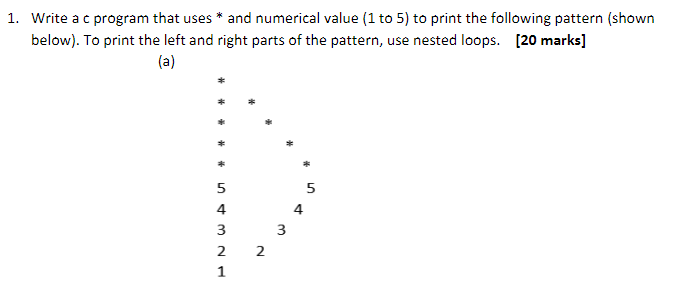
# Lab 21+22

# Lab FAT

# Submitted by: Alokam Nikhitha

# Reg No:19BCE2555

# QUESTION 1:



# CODE:

**#include<stdio.h>**

**int main() {**

**printf("\n \n \*\* 19BCE2555 \*\*\* \n \n");**

**int rows=5;**

**for(int i=1;i<=rows;++i) {**

**for(int j=1;j<=i;++j) {**

**if(j==1 || j==i) printf("\* ");**

**else printf(" ");**

**}**

**printf("\n");**

**}**

**for(int i=rows;i>=0;--i) {**

**for(int j=i;j>=1;--j) {**

**if(j==1||j==i) printf("%d ", i);**

**else printf(" ");**

**}**

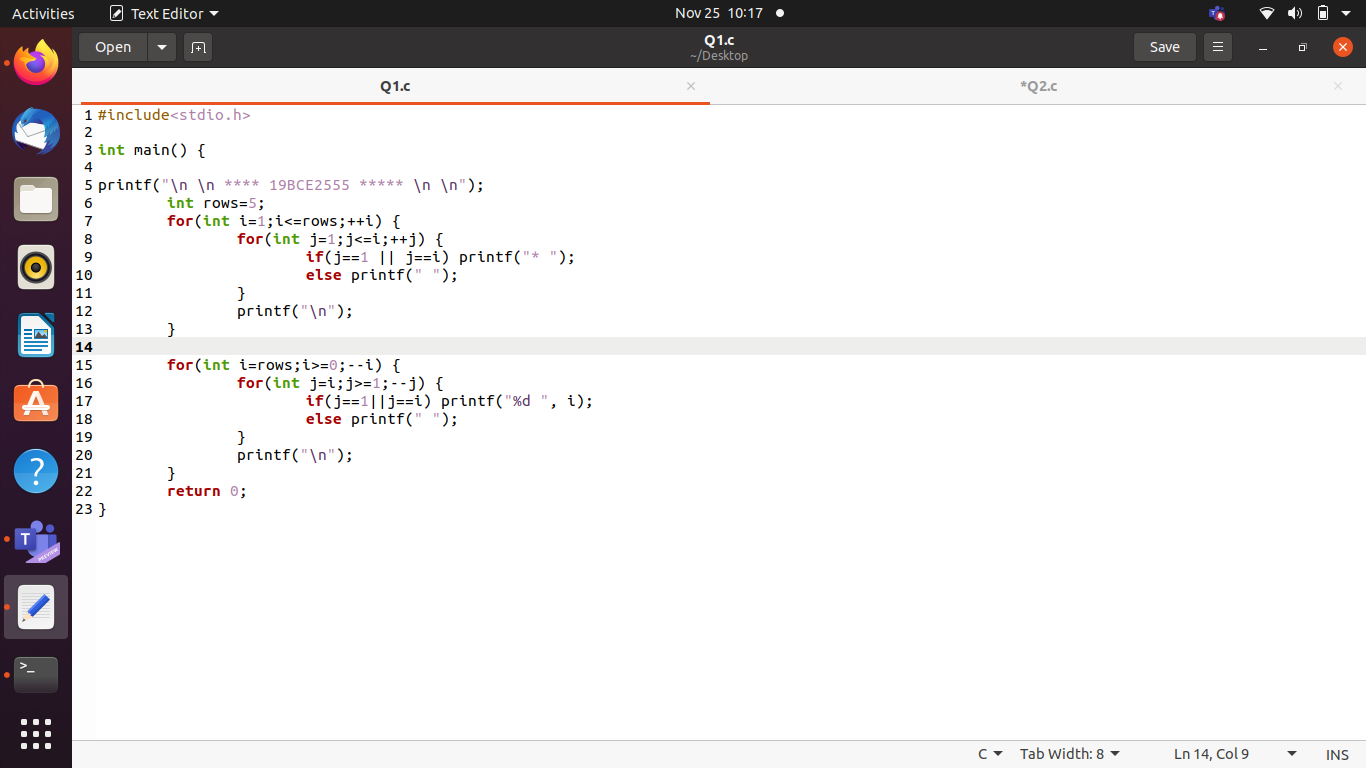
**printf("\n");**

**}**

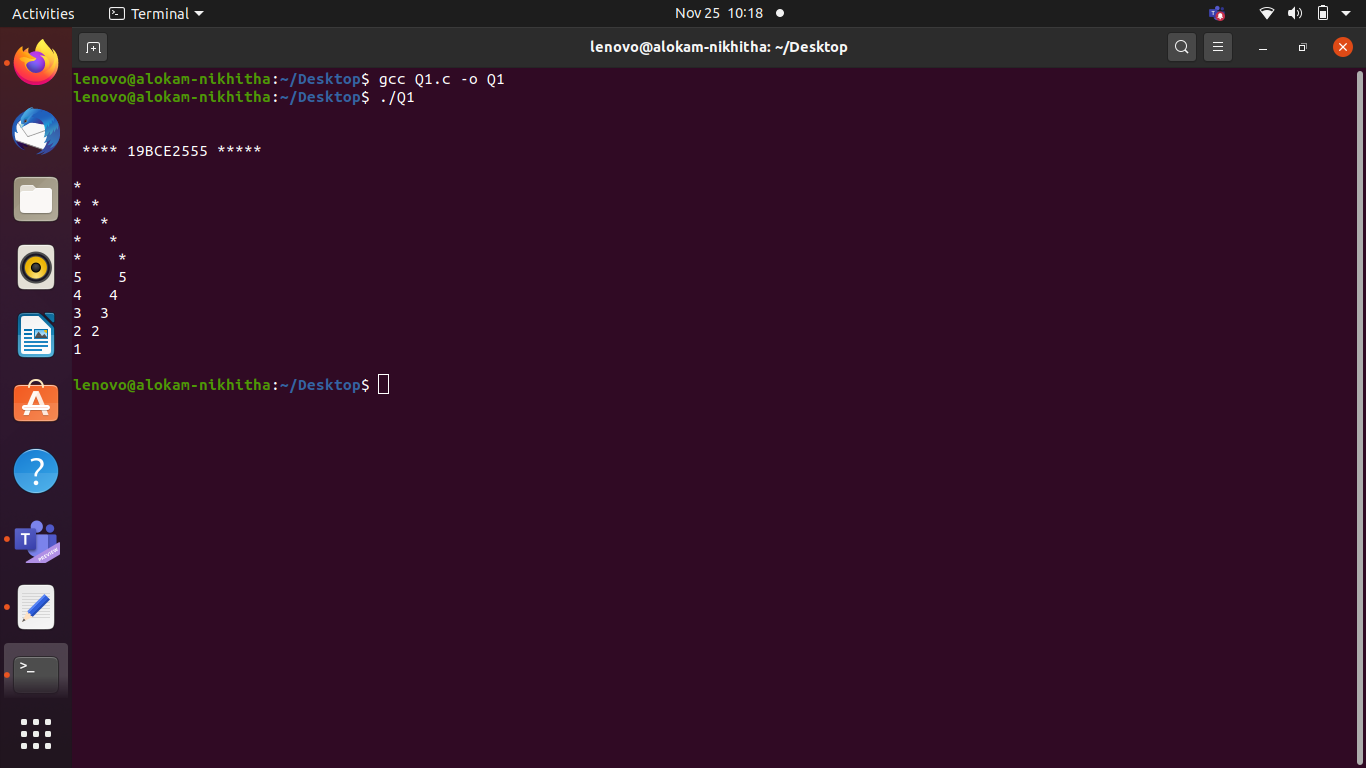
**return 0;**

**}**

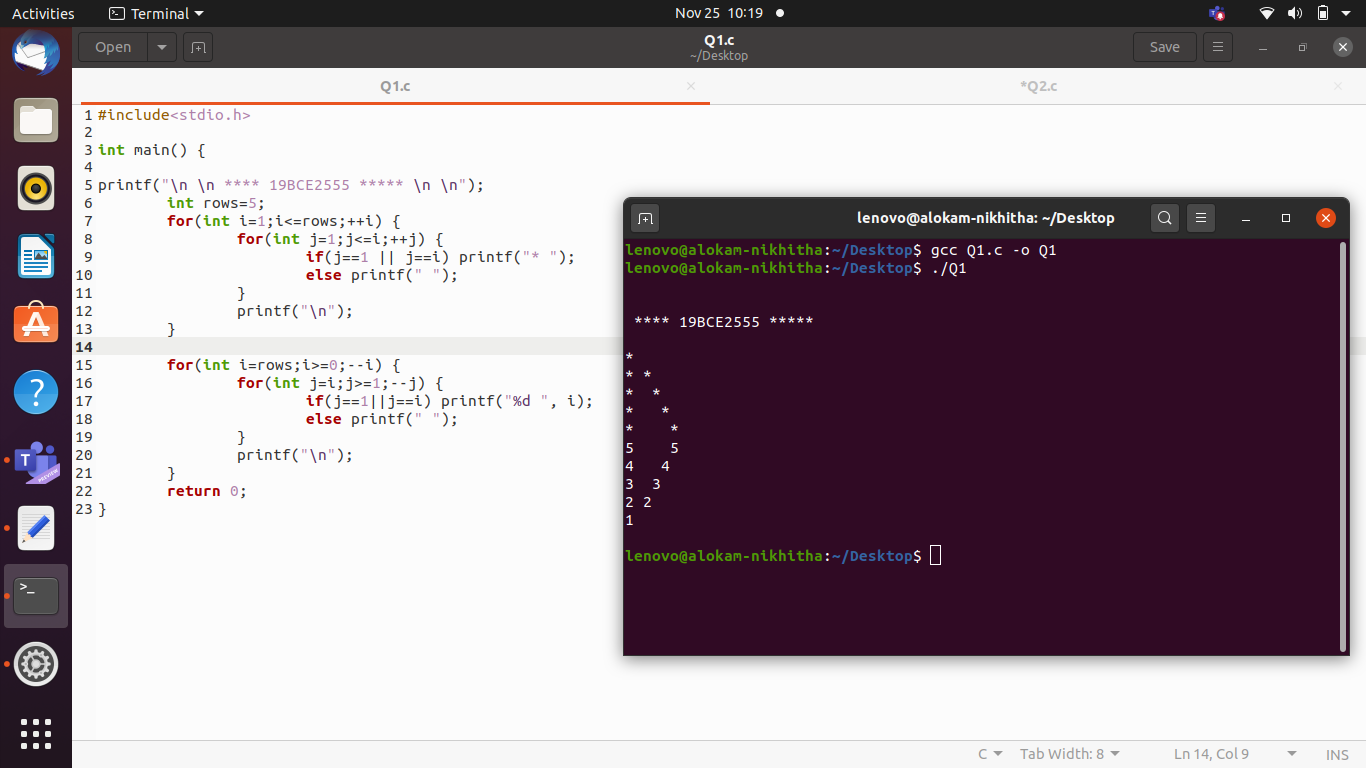
# Code Snippets:



# OUTPUT:



**OUTPUT WITH CODE:**



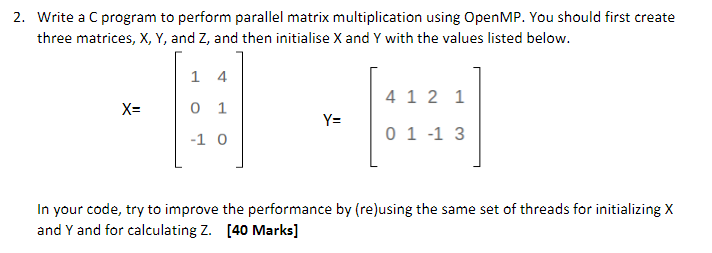
Result and Inferences:

- Nested loops are used in order to print the pattern.

- First loop prints the upper half of the pattern which prints \* predominantly.

- Second loop prints the lower half of the patter where it prints numbers from 5 to 1 in decreasing order.

# QUESTION 2:



# CODE:

#include <stdio.h>

#include <time.h>

#include <omp.h>

#include <stdlib.h>

int X[3][2]={{1, 4}, {0, 1}, {-1, 0}};

int Y[2][4]={{4, 1, 2, 1}, {0, 1, -1, 3}};

int Z[3][4];

int main ()

{

int i, j, k;

printf ("19BCE2555\n");

double start, end;

start = omp\_get\_wtime ();

#pragma omp parallel for private(i,j,k) shared(X,Y,Z)

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 4; j++)

{

Z[i][j]=0;

for (int k = 0; k < 2; k++)

Z[i][j] += X[i][k] \* Y[k][j];

}

}

printf ("Matrix X\n");

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 2; j++)

printf ("%d\t", X[i][j]);

printf ("\n");

}

printf ("\n");

printf ("Matrix Y\n");

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 4; j++)

printf ("%d\t", Y[i][j]);

printf ("\n");

}

printf ("\n");

printf ("Resultant Matrix for Mutliplication\n");

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 4; j++)

printf ("%d\t", Z[i][j]);

printf ("\n");

}

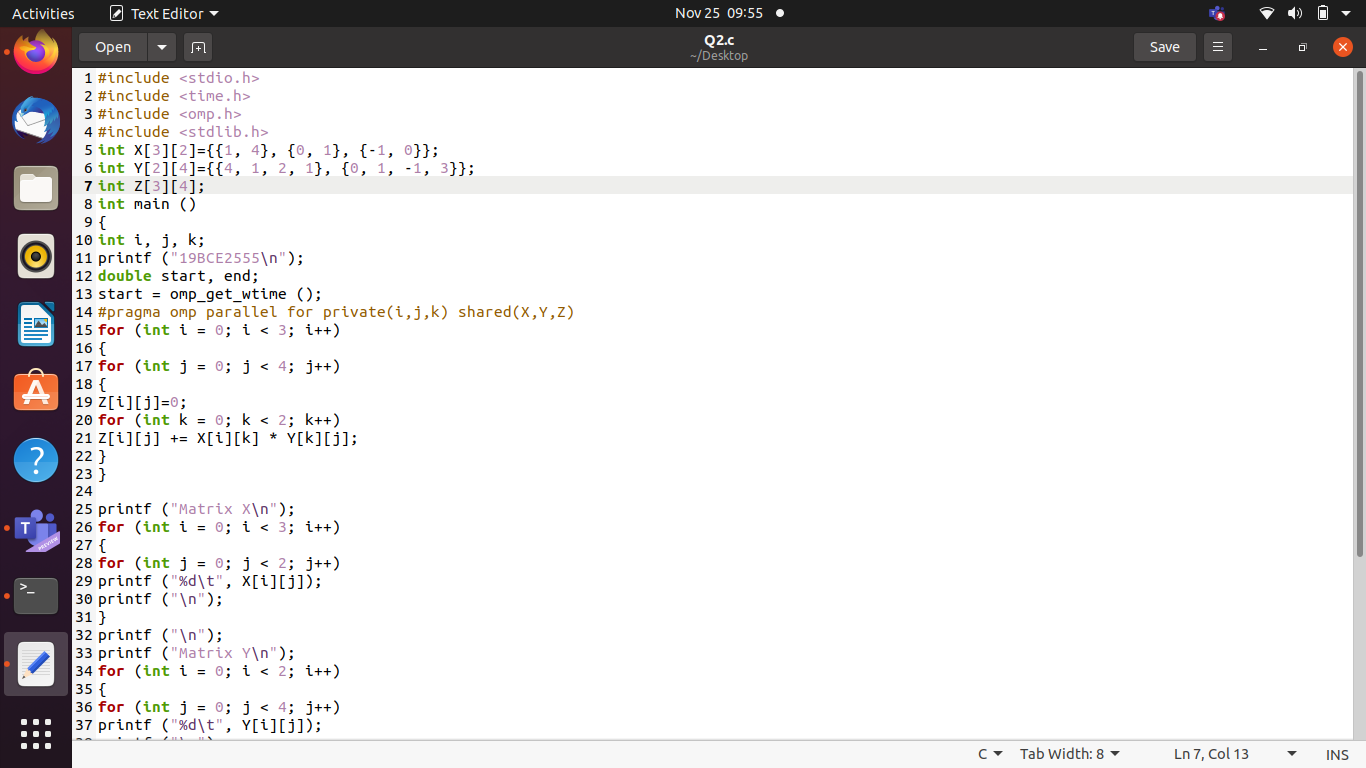
end = omp\_get\_wtime ();

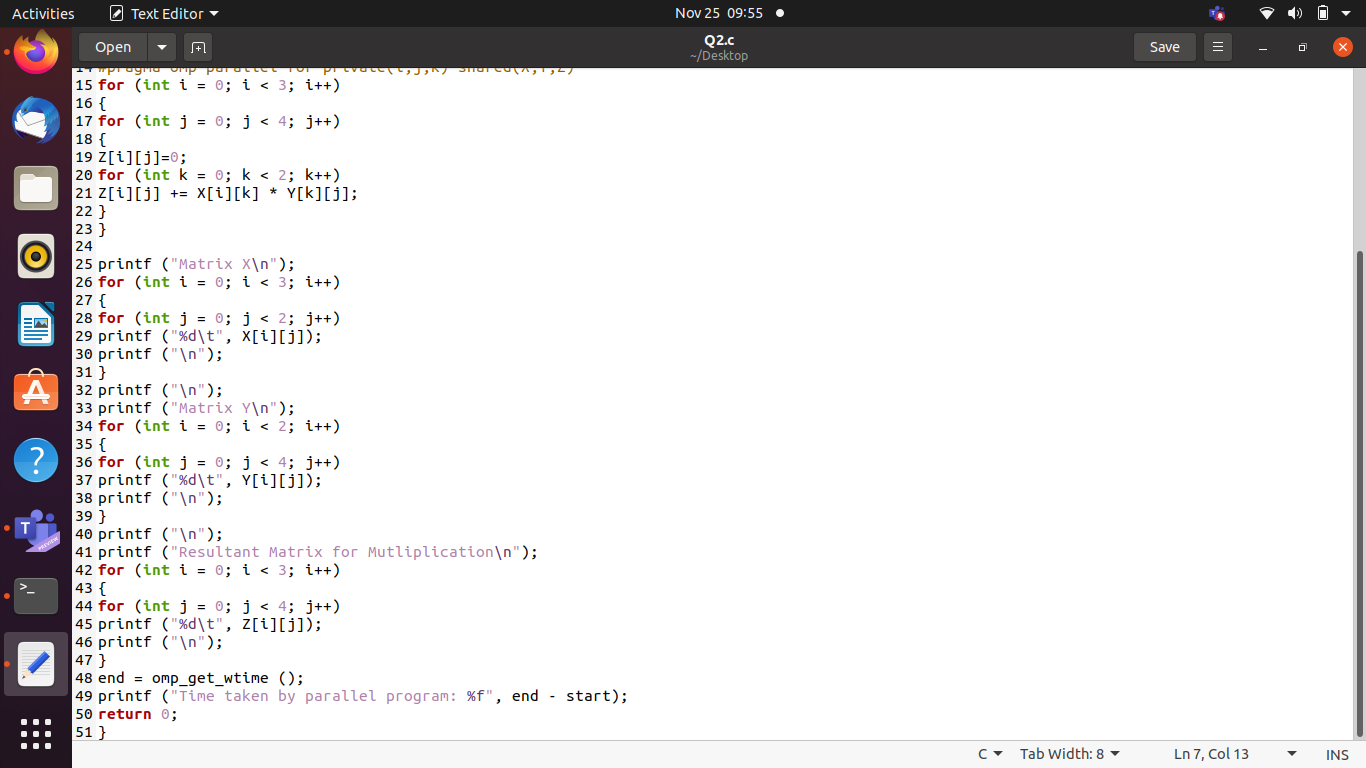
printf ("Time taken by parallel program: %f", end - start);

return 0;

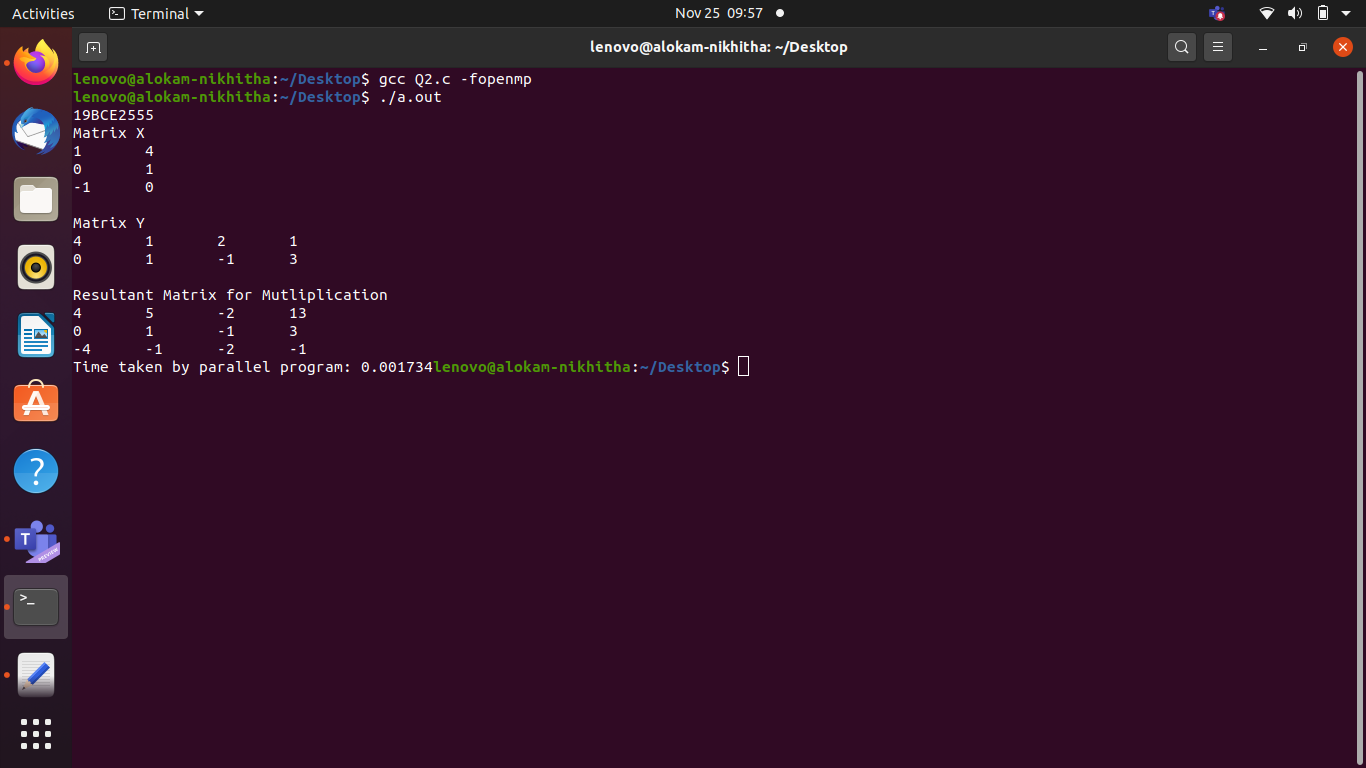
}

# Code Snippets:

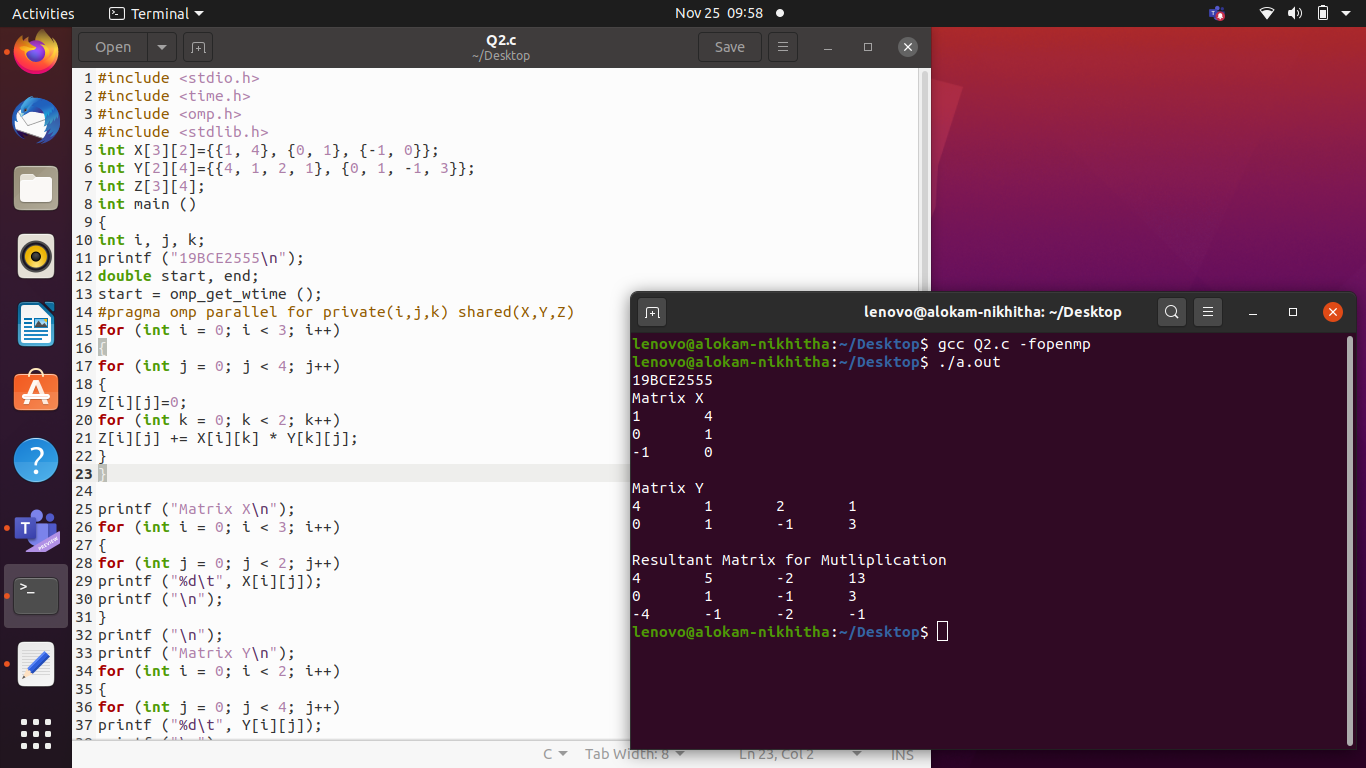




# OUTPUT:



**OUTPUT WITH CODE:**



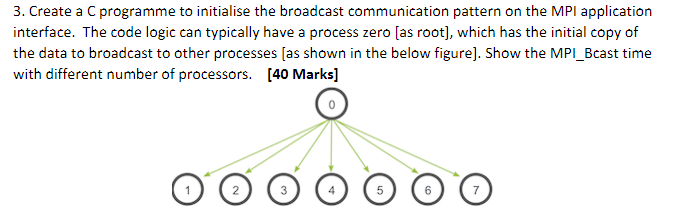
Result and Inferences:

- Here we have initialized our X, Y and Z matrices globally for global access.

- We have then multiplied them parallelly using OpenMP.

- Time taken by parallel program to execute the same is 0.001734 milliseconds (ms).

# QUESTION 3:



# CODE:

#include<stdio.h>

#include<stdlib.h>

#include<mpi.h>

int main(int argc, char\*argv[] ) {

MPI\_Init(&argc, &argv);

int my\_rank;

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &my\_rank);

int broadcast\_root=0;

int buffer;

if(my\_rank==broadcast\_root) {

buffer=192555;

printf("Process %d sending data %d\n", my\_rank, buffer);

}

MPI\_Bcast(&buffer,1,MPI\_INT, broadcast\_root, MPI\_COMM\_WORLD);

if(my\_rank!=broadcast\_root) {

printf("Process %d recieveing via broadcast, data %d\n", my\_rank, buffer);

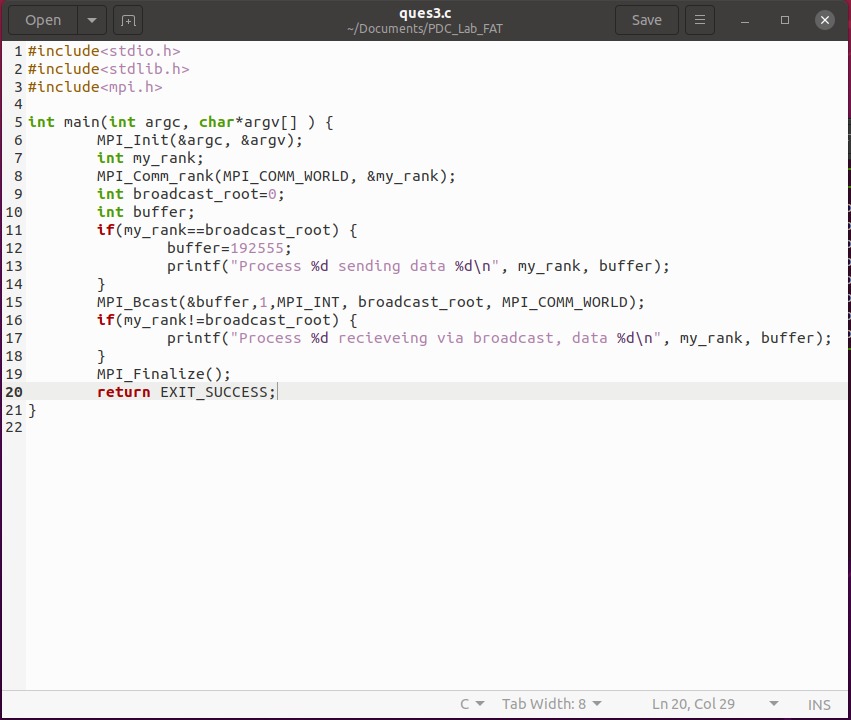
}

MPI\_Finalize();

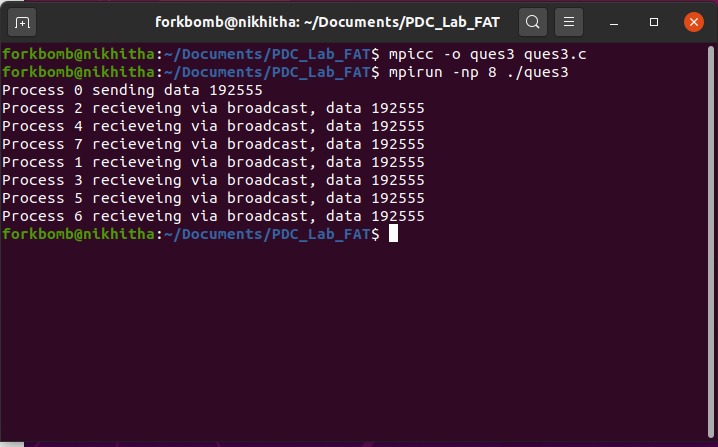
return EXIT\_SUCCESS;

}

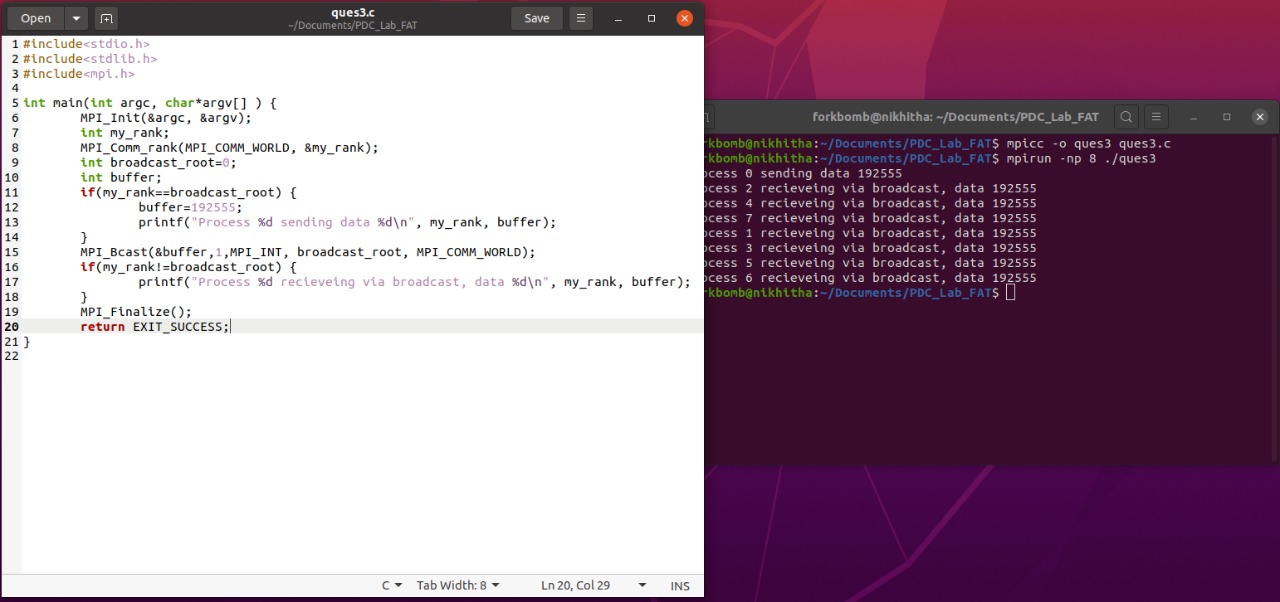
# Code Snippets:



# OUTPUT:



**OUTPUT WITH CODE:**



Result and Inferences:

- We used Broadcast method of mpi to demonstrate the broadcasting of a message.

- The message broadcasted is 192555, (corresponding to registration number 19BCE2555).

- We can see that the process 0 is broadcasting the above number. All the other processes numbered from 1 to 7 are receiving the broadcasted message 192555.