Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

# CSE4001 Parallel and Distributed Computing

## Digital Assignment-3 (ELA)

**Name: Nakul Jadeja**

**Reg no.: 19BCE0660**

**School of Computer Science and Engineering**

**Course Code: CSE4001**

**Slot: L11+L12**

**Winter Semester 2021-22**

**Professor: Narayanamoorthi M**

**Question: -**

***1) Write a c program using open MP to perform the following matrix operations. (Vary the size of the matrix like 3x3, 5x5, 10x10, 20x20, 30x30 and thread numbers and observe the time to compute for serial and parallel execution)***

***(i) Matrix Addition***

***(ii) Matrix subtraction***

***(iii) Matrix multiplication***

***(iv) To find the sum of rows and column elements of the matrix***

***(v) To print the lower triangle and upper triangle matrix***

***(vi) Write a parallel algorithm to transpose the matrix nxn***

***i) <u>Matrix Addition Code:</u>***

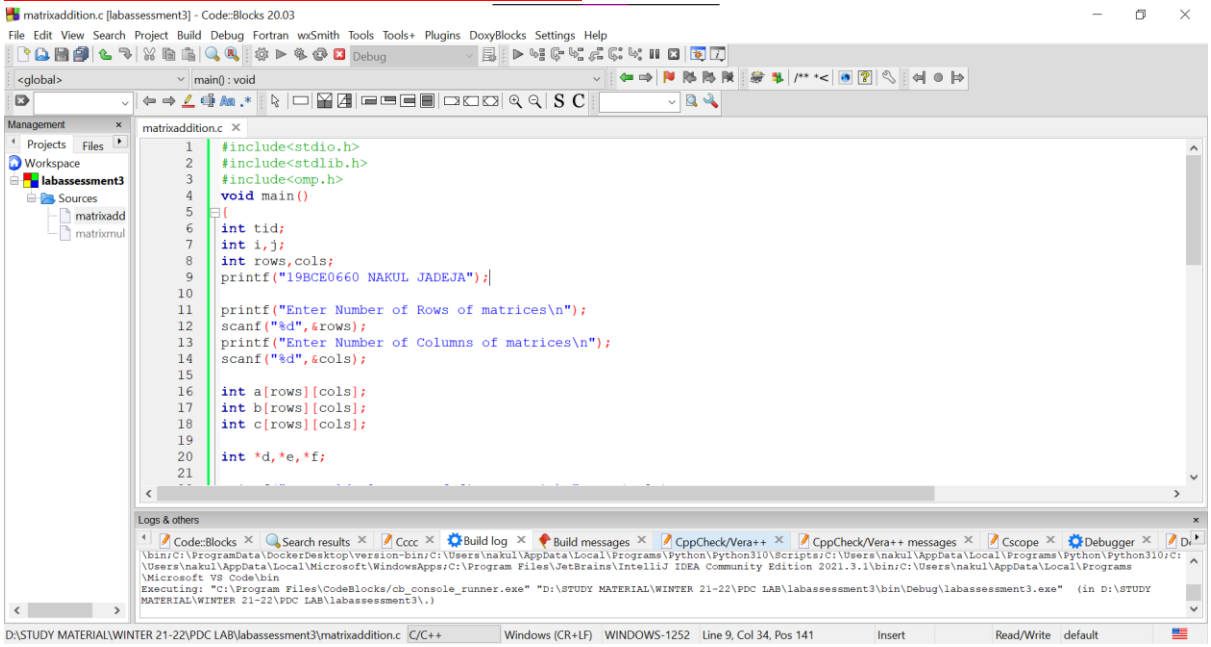
***<u>CODE:-</u>***

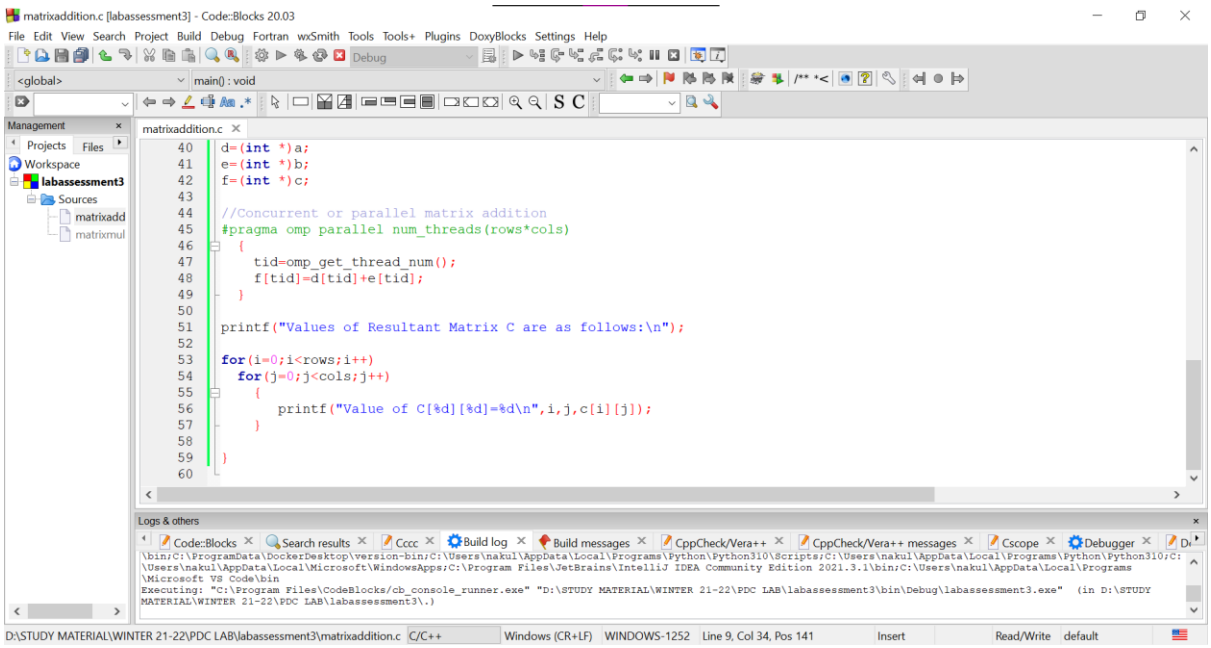
```
#include<stdio.h>
#include<stdlib.h>
#include<omp.h>
void main()
{
int tid;
int i,j;
int rows,cols;

printf("Enter Number of Rows of matrices\n");
```

```
scanf("%d",&rows);
printf("Enter Number of Columns of matrices\n");
scanf("%d",&cols);

int a[rows][cols];
int b[rows][cols];
int c[rows][cols];

int *d,*e,*f;

printf("Enter %d elements of first matrix\n",rows*cols);
for(i=0;i<rows;i++)
  for(j=0;j<cols;j++)
    {
      scanf("%d",&a[i][j]);
    }

printf("Enter %d elements of second matrix\n",rows*cols);
for(i=0;i<rows;i++)
  for(j=0;j<cols;j++)
    {
      scanf("%d",&b[i][j]);
    }

d=(int *)malloc(sizeof(int)*rows*cols);
e=(int *)malloc(sizeof(int)*rows*cols);
f=(int *)malloc(sizeof(int)*rows*cols);
```

```
d=(int *)a;
e=(int *)b;
f=(int *)c;

//Concurrent or parallel matrix addition
#pragma omp parallel num_threads(rows*cols)
  {
    tid=omp_get_thread_num();
    f[tid]=d[tid]+e[tid];
  }

printf("Values of Resultant Matrix C are as follows:\n");

for(i=0;i<rows;i++)
  for(j=0;j<cols;j++)
    {
       printf("Value of C[%d][%d]=%d\n",i,j,c[i][j]);
    }

}
```

# SCREENSHOT OF THE OUTPUT:-



```
"D:\STUDY MATERIAL\WINTER 21-22\PDC LAB\labassessment3\bin\Debug\labassessment3.exe"
Enter Number of Rows of matrices
2
Enter Number of Columns of matrices
3
Enter 6 elements of first matrix
1 2 3 4 5 6
Enter 6 elements of second matrix
2 3 4 5 6 7
Values of Resultant Matrix C are as follows:
Value of C[0][0]=3
Value of C[0][1]=5
Value of C[0][2]=7
Value of C[1][0]=9
Value of C[1][1]=11
Value of C[1][2]=13

Process returned 2 (0x2)   execution time : 10.414 s
Press any key to continue.
```

```
"D:\STUDY MATERIAL\WINTER 21-22\PDC LAB\labassessment3\bin\Debug\labassessment3.exe"
19BCE0660 NAKUL JADEJAEnter Number of Rows of matrices
2
Enter Number of Columns of matrices
3
Enter 6 elements of first matrix
1 2 3 4 5 6
Enter 6 elements of second matrix
2 3 4 5 6 7
Values of Resultant Matrix C are as follows:
Value of C[0][0]=3
Value of C[0][1]=5
Value of C[0][2]=7
Value of C[1][0]=9
Value of C[1][1]=11
Value of C[1][2]=13

Process returned 2 (0x2)   execution time : 11.020 s
Press any key to continue.
```

## ii) MATRIX SUBTRACTION: -

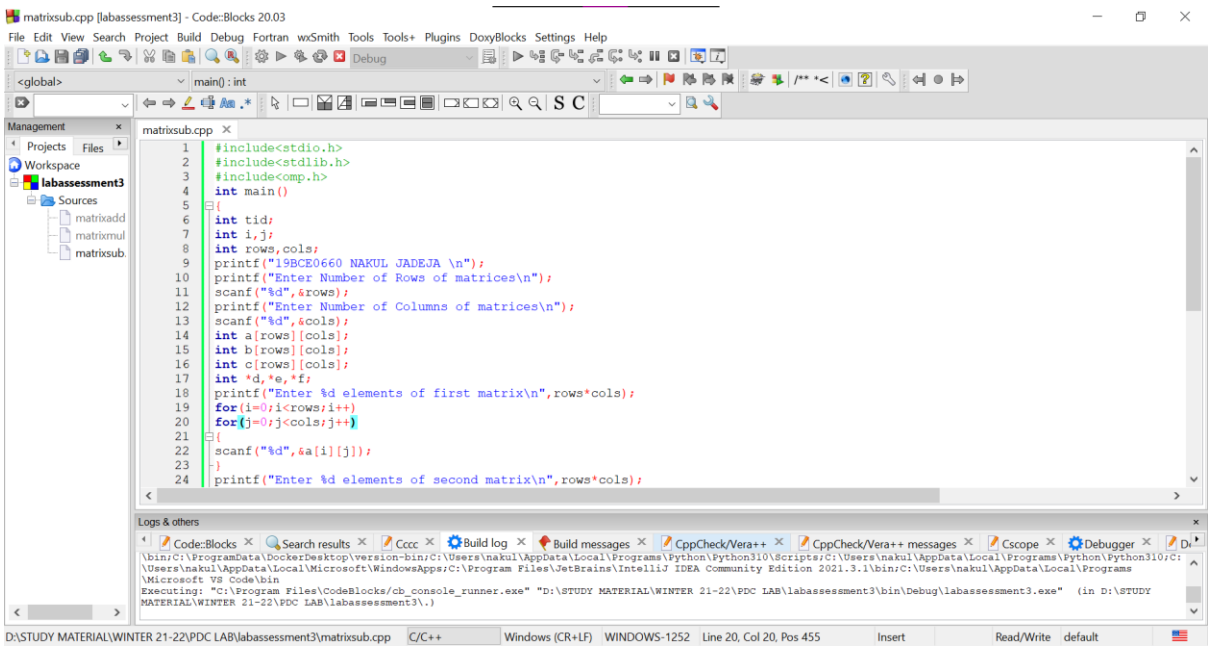
**CODE: -**

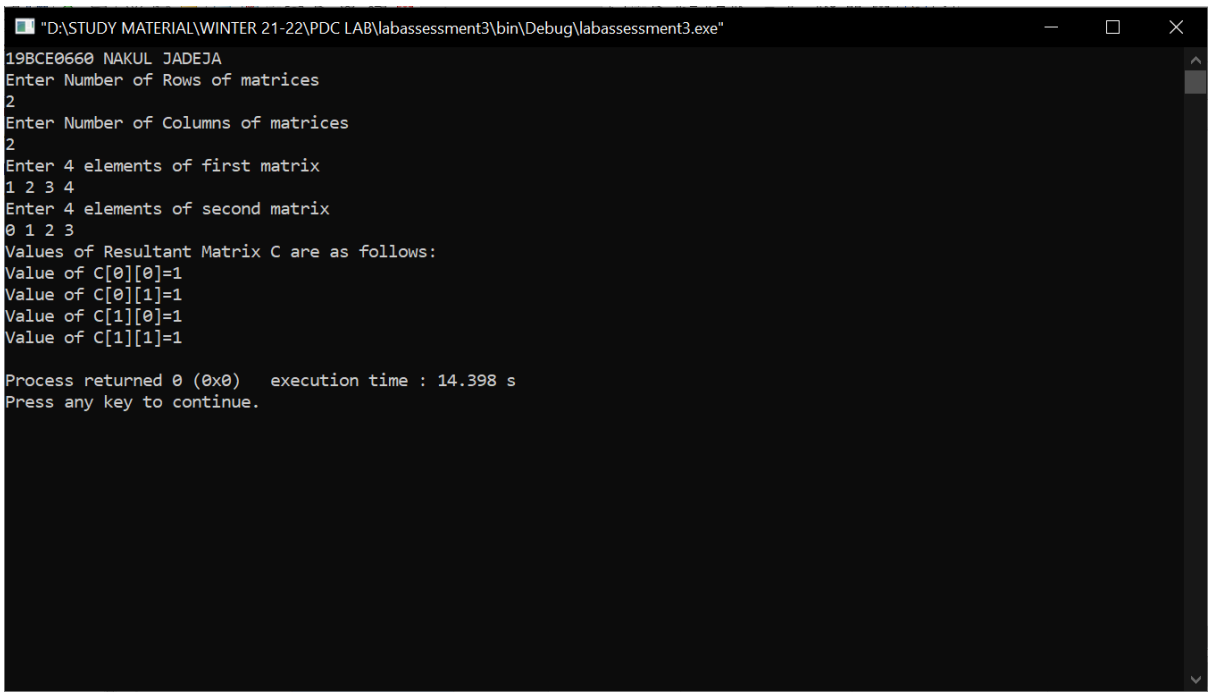
```
#include<stdio.h>

#include<stdlib.h>
```

```
#include<omp.h>
int main()
{
int tid;
int i,j;
int rows,cols;
printf("19BCE0660 NAKUL JADEJA \n");
printf("Enter Number of Rows of matrices\n");
scanf("%d",&rows);
printf("Enter Number of Columns of matrices\n");
scanf("%d",&cols);
int a[rows][cols];
int b[rows][cols];
int c[rows][cols];
int *d,*e,*f;
printf("Enter %d elements of first matrix\n",rows*cols);
for(i=0;i<rows;i++)
for(j=0;j<cols;j++)
{
scanf("%d",&a[i][j]);
}
printf("Enter %d elements of second matrix\n",rows*cols);
for(i=0;i<rows;i++)
for(j=0;j<cols;j++)
{
scanf("%d",&b[i][j]);
}
```

```
d=(int *)malloc(sizeof(int)*rows*cols);
e=(int *)malloc(sizeof(int)*rows*cols);
f=(int *)malloc(sizeof(int)*rows*cols);
d=(int *)a;
e=(int *)b;
f=(int *)c;
//Concurrent or parallel matrix addition
#pragma omp parallel num_threads(rows*cols)
{
tid=omp_get_thread_num();
f[tid]=d[tid]-e[tid];
}
printf("Values of Resultant Matrix C are as follows:\n");
for(i=0;i<rows;i++)
for(j=0;j<cols;j++)
{
printf("Value of C[%d][%d]=%d\n",i,j,c[i][j]);
}
}
```

***<u>SCREENSHOT OF THE OUTPUT: -</u>***




### *iii)* ***MATRIX MULTIPLICATION: -***

***CODE: -***

```
#include <stdio.h>

#include <stdlib.h>

#include <omp.h>

#include <sys/time.h>
```

```
#define N 30

int A[N][N];
int B[N][N];
int C[N][N];

int main()
{
    int i,j,k;
    struct timeval tv1, tv2;
    struct timezone tz;
          double elapsed;
    omp_set_num_threads(omp_get_num_procs());
    for (i= 0; i< N; i++)
        for (j= 0; j< N; j++)
          {
            A[i][j] = 2;
            B[i][j] = 2;
          }
    gettimeofday(&tv1, &tz);
    #pragma omp parallel for private(i,j,k) shared(A,B,C)
    for (i = 0; i < N; ++i) {
        for (j = 0; j < N; ++j) {
            for (k = 0; k < N; ++k) {
                C[i][j] += A[i][k] * B[k][j];
```

```
            }
        }
    }



    gettimeofday(&tv2, &tz);
    elapsed = (double) (tv2.tv_sec-tv1.tv_sec) + (double) (tv2.tv_usec-
tv1.tv_usec) * (1.e-6);
    printf("elapsed time = %f seconds.\n", elapsed);



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

***SCREENSHOT OF THE OUTPUT:-***

matrixmul.c [labassessment3] - Code::Blocks 20.03

```c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <sys/time.h>


#define N 30

int A[N][N];
int B[N][N];
int C[N][N];

int main()
{
    int i,j,k;
    struct timeval tv1, tv2;
    struct timezone tz;
    double elapsed;
    omp_set_num_threads(omp_get_num_procs());
    for (i= 0; i< N; i++)
        for (j= 0; j< N; j++)
    {
            A[i][j] = 2;
            B[i][j] = 2;
```

Logs & others

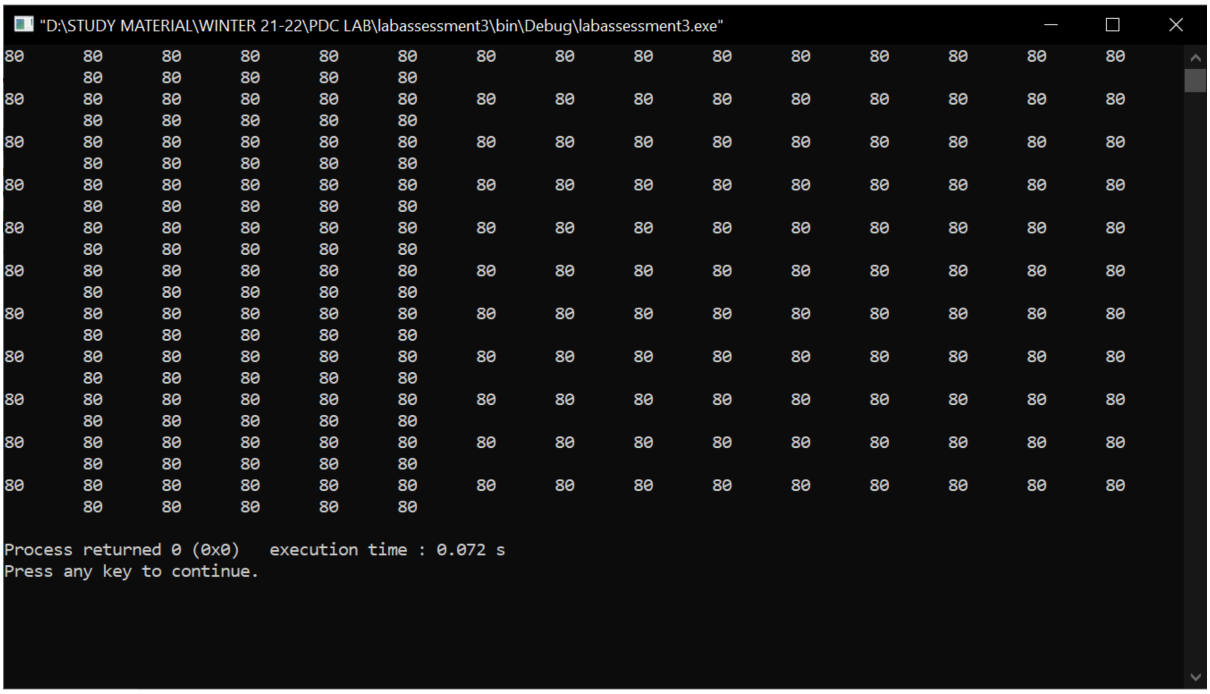
```
\Microsoft VS Code\bin
Executing: "C:\Program Files\CodeBlocks/cb_console_runner.exe" "D:\STUDY MATERIAL\WINTER 21-22\PDC LAB\labassessment3\bin\Debug\labassessment3.exe"  (in D:\STUDY MATERIAL\WINTER 21-22\PDC LAB\labassessment3\.)
Process terminated with status 0 (0 minute(s), 12 second(s))
```

"D:\STUDY MATERIAL\WINTER 21-22\PDC LAB\labassessment3\bin\Debug\labassessment3.exe"

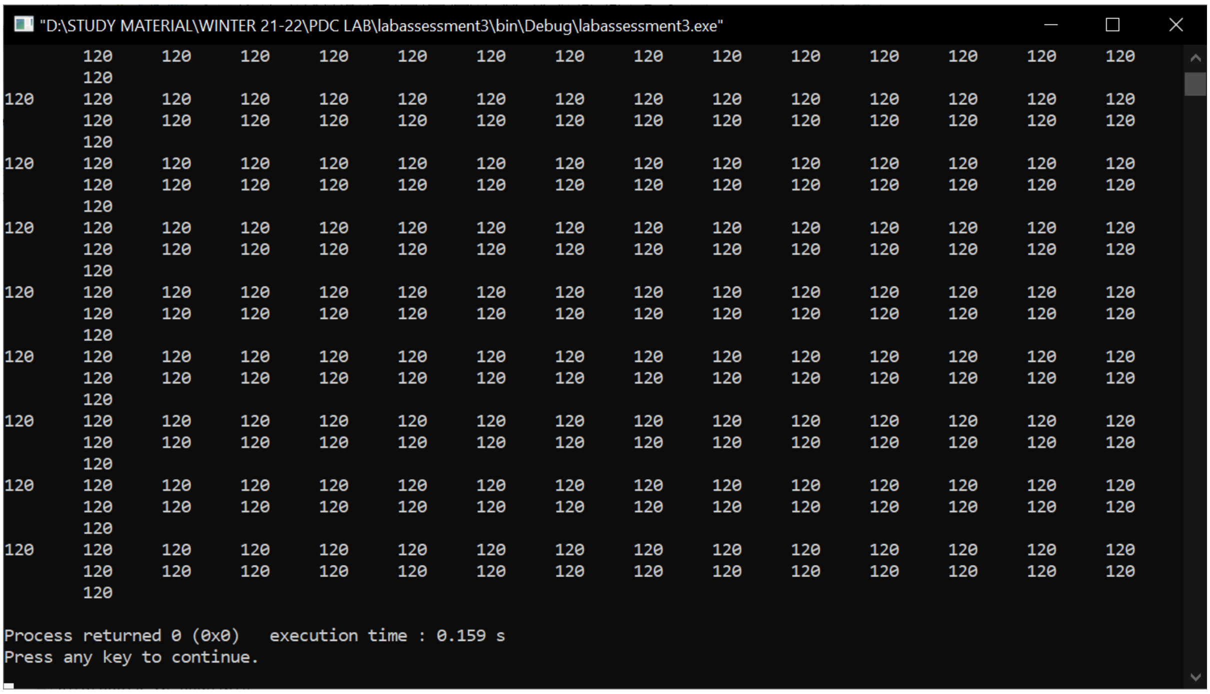
```
elapsed time = 0.003991 seconds.
12      12      12
12      12      12
12      12      12

Process returned 0 (0x0)   execution time : 0.028 s
Press any key to continue.
```

```
"D:\STUDY MATERIAL\WINTER 21-22\PDC LAB\labassessment3\bin\Debug\labassessment3.exe"
elapsed time = 0.000000 seconds.
20      20      20      20      20
20      20      20      20      20
20      20      20      20      20
20      20      20      20      20
20      20      20      20      20

Process returned 0 (0x0)   execution time : 0.020 s
Press any key to continue.
```

```
"D:\STUDY MATERIAL\WINTER 21-22\PDC LAB\labassessment3\bin\Debug\labassessment3.exe"
elapsed time = 0.000000 seconds.
40      40      40      40      40      40      40      40      40      40
40      40      40      40      40      40      40      40      40      40
40      40      40      40      40      40      40      40      40      40
40      40      40      40      40      40      40      40      40      40
40      40      40      40      40      40      40      40      40      40
40      40      40      40      40      40      40      40      40      40
40      40      40      40      40      40      40      40      40      40
40      40      40      40      40      40      40      40      40      40
40      40      40      40      40      40      40      40      40      40
40      40      40      40      40      40      40      40      40      40

Process returned 0 (0x0)   execution time : 0.035 s
Press any key to continue.
```



```
80      80      80      80      80      80      80      80      80      80      80      80      80      80      80
        80      80      80      80      80
80      80      80      80      80      80      80      80      80      80      80      80      80      80      80
        80      80      80      80      80
80      80      80      80      80      80      80      80      80      80      80      80      80      80      80
        80      80      80      80      80
80      80      80      80      80      80      80      80      80      80      80      80      80      80      80
        80      80      80      80      80
80      80      80      80      80      80      80      80      80      80      80      80      80      80      80
        80      80      80      80      80
80      80      80      80      80      80      80      80      80      80      80      80      80      80      80
        80      80      80      80      80
80      80      80      80      80      80      80      80      80      80      80      80      80      80      80
        80      80      80      80      80
80      80      80      80      80      80      80      80      80      80      80      80      80      80      80
        80      80      80      80      80
80      80      80      80      80      80      80      80      80      80      80      80      80      80      80
        80      80      80      80      80
80      80      80      80      80      80      80      80      80      80      80      80      80      80      80
        80      80      80      80      80
80      80      80      80      80      80      80      80      80      80      80      80      80      80      80
        80      80      80      80      80

Process returned 0 (0x0)   execution time : 0.072 s
Press any key to continue.
```



```
        120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120
120     120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120
120     120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120
120     120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120
120     120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120
120     120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120
120     120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120
120     120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120
120     120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120     120     120     120     120     120     120     120     120     120     120     120     120     120
        120

Process returned 0 (0x0)   execution time : 0.159 s
Press any key to continue.
```

## *iv) SUM OF ROW AND COLUMN ELEMENTS: -*

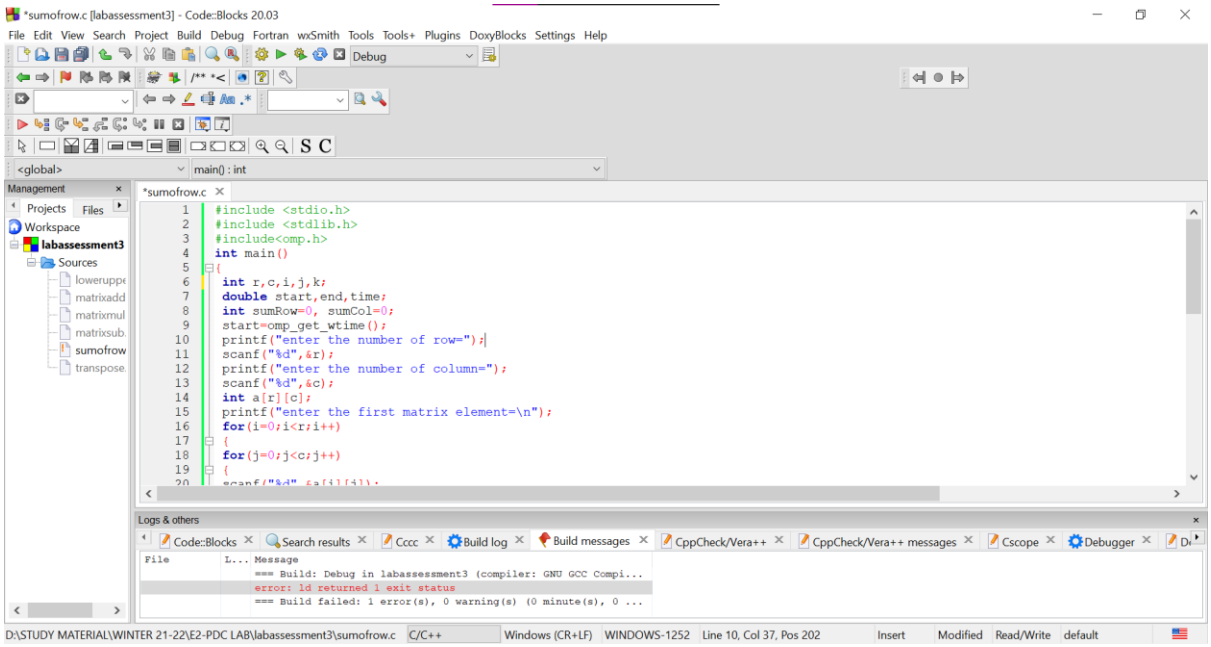
**CODE: -**

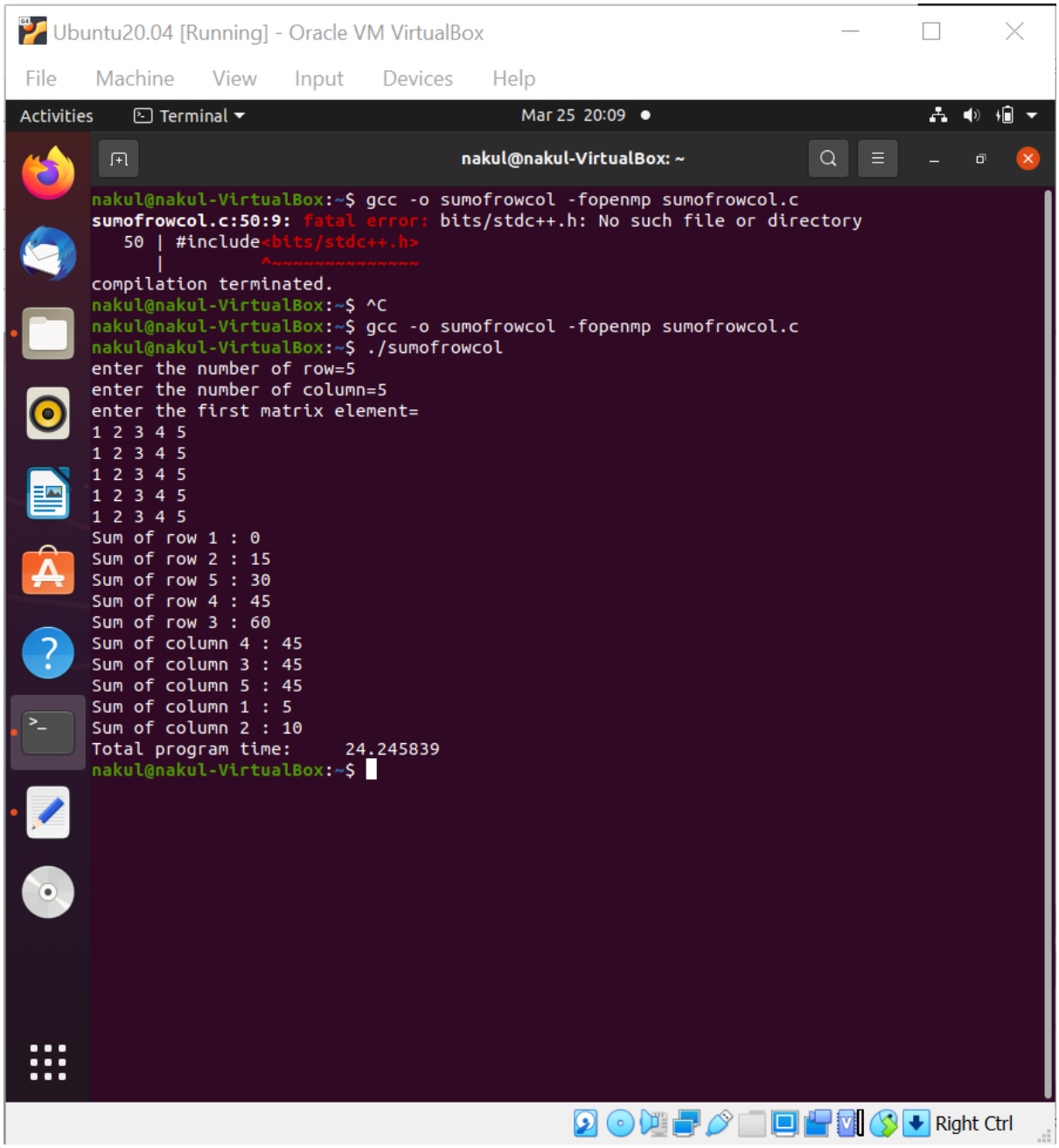
```
#include <stdio.h>

#include <stdlib.h>

#include<omp.h>
```

```
int main()
{
 int r,c,i,j,k;
 double start,end,time;
 int sumRow=0, sumCol=0;
 start=omp_get_wtime();
 printf("enter the number of row=");
 scanf("%d",&r);
 printf("enter the number of column=");
 scanf("%d",&c);
 int a[r][c];
 printf("enter the first matrix element=\n");
 for(i=0;i<r;i++)
 {
 for(j=0;j<c;j++)
 {
 scanf("%d",&a[i][j]);
 }
 }
 #pragma omp parallel for
 for(int i = 0; i < r; i++)
 {
 sumRow = 0;
 #pragma omp parallel for
 for(int j = 0; j < c; j++)
 {
 sumRow = sumRow + a[i][j];
```

```
}
printf("Sum of row %d : %d\n", (i+1), sumRow);
}
#pragma omp parallel for
for(int i = 0; i < c; i++)
{
sumCol = 0;
#pragma omp parallel for
for(int j = 0; j < r; j++)
{
sumCol = sumCol + a[j][i];
}
printf("Sum of column %d : %d\n", (i+1), sumCol);
}
end=omp_get_wtime();
time=end-start;
printf("Total program time:\t%f\n",time);
return 0;
}
```

***SCREENSHOT OF THE OUTPUT: -***


*sumofrow.c [labassessment3] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
<global>
main() : int
Management
Projects Files
Workspace
labassessment3
Sources
lowerupp
matrixadd
matrixmul
matrixsub
sumofrow
transpose
*sumofrow.c
#include <stdio.h>
#include <stdlib.h>
#include<omp.h>
int main()
{
int r,c,i,j,k;
double start,end,time;
int sumRow=0, sumCol=0;
start=omp_get_wtime();
printf("enter the number of row=");
scanf("%d",&r);
printf("enter the number of column=");
scanf("%d",&c);
int a[r][c];
printf("enter the first matrix element=\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck/Vera++
CppCheck/Vera++ messages
Cscope
Debugger
File
L...
Message
=== Build: Debug in labassessment3 (compiler: GNU GCC Compi...
error: ld returned 1 exit status
=== Build failed: 1 error(s), 0 warning(s) (0 minute(s), 0 ...
D:\STUDY MATERIAL\WINTER 21-22\E2-PDC LAB\labassessment3\sumofrow.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 10, Col 37, Pos 202
Insert
Modified
Read/Write
default




## *v) LOWER-UPPER TRIANGULAR MATRIX: -*

***CODE: -***

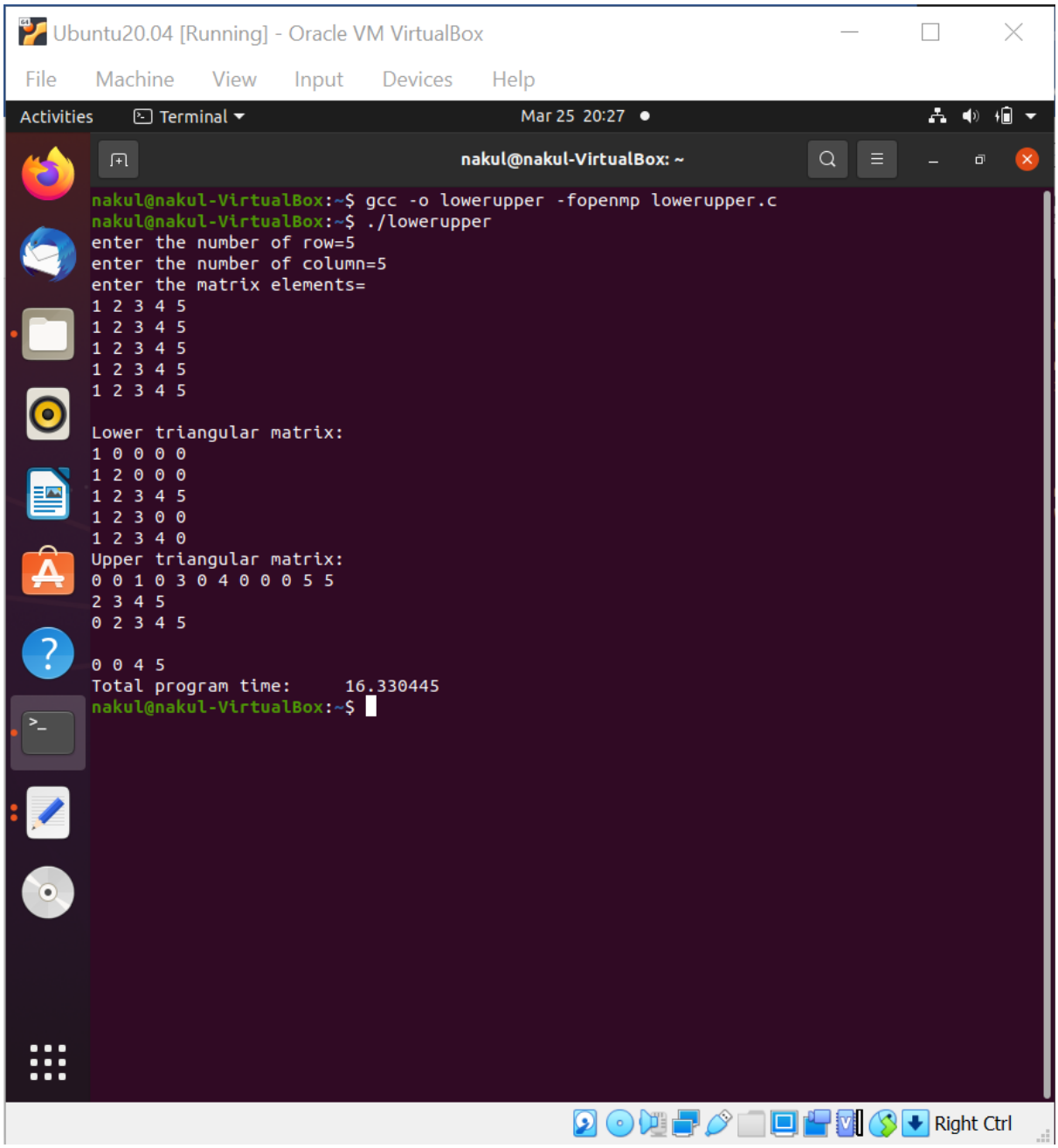
```
#include <stdio.h>

#include <stdlib.h>

#include<omp.h>

int main()
```

```
{
 double start,end,time;
 start=omp_get_wtime();
 int row,column,i,j;
 printf("enter the number of row=");
 scanf("%d",&row);
 printf("enter the number of column=");
 scanf("%d",&column);
 int matrix[row][column];
 printf("enter the matrix elements=\n");
 for(i=0;i<row;i++)
 {
 for(j=0;j<column;j++)
 {
 scanf("%d",&matrix[i][j]);
 }
 }
 printf("\n");
 // printing lower triangular matrix
 printf("Lower triangular matrix: \n");
 #pragma omp parallel for
 for (int i = 0; i < row; i++)
 {
 #pragma omp parallel for
 for (int j = 0; j < column; j++)
 {
 if (i < j)
```

```
 {
printf("0 ");
 }
 else
    printf("%d ",matrix[i][j]);
 }
 printf("\n");
 }
 // printing upper triangular matrix
 printf("Upper triangular matrix: \n");
 #pragma omp parallel for
 for (int i = 0; i < row; i++)
 {
 #pragma omp parallel for
 for (int j = 0; j < column; j++)
 {
 if (i > j)
 {
 printf("0 ");
 }
 else
 printf("%d ",matrix[i][j]);
 }
 printf("\n");
 }
 end=omp_get_wtime();
 time=end-start;
```

```
  printf("Total program time:\t%f\n",time);
 return 0;
}
```

***SCREENSHOT OF THE OUTPUT: -***




***vi) TRANSPOSE OF MATRIX: -***

***CODE: -***

```
#include <stdio.h>
#include <sys/time.h>
#include <omp.h>

/* Main Program */

main()
{
        int             NoofRows, NoofCols, i, j;
        float         **Matrix, **Trans, **Checkoutput, flops;

        struct timeval  TimeValue_Start;
        struct timezone TimeZone_Start;

        struct timeval  TimeValue_Final;
        struct timezone TimeZone_Final;
        long            time_start, time_end, time_overhead;

    printf("19BCE0660 NAKUL JADEJA");
        printf("Read The Matrix Size Noofrows And Colums Of Matrix \n");
        scanf("%d%d", &NoofRows, &NoofCols);

        if (NoofRows <= 0 || NoofCols <= 0) {
                printf("The NoofRows And NoofCols Should Be Of Positive
Sign\n");
                exit(1);
        }
        /* Matrix Elements */
```

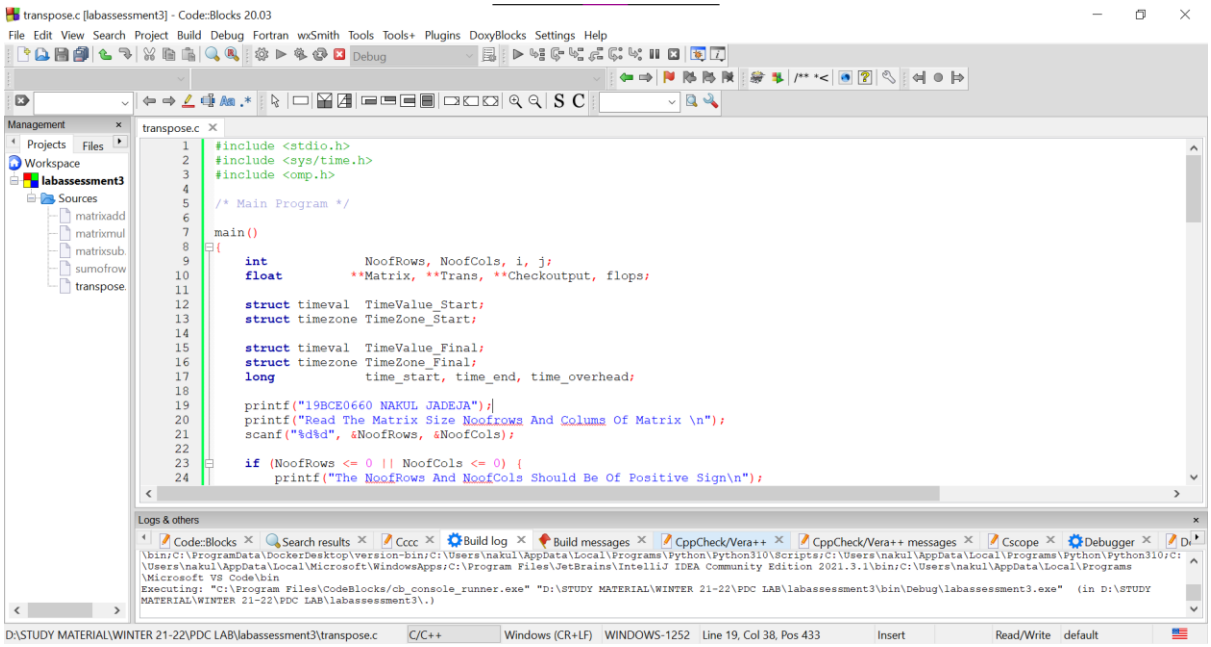
```
Matrix = (float **) malloc(sizeof(float) * NoofRows);
for (i = 0; i < NoofRows; i++) {
	Matrix[i] = (float *) malloc(sizeof(float) * NoofCols);
	for (j = 0; j < NoofCols; j++)
		Matrix[i][j] = (i * j) * 5 + i;
}

/* Dynamic Memory Allocation */

Trans = (float **) malloc(sizeof(float) * NoofCols);
Checkoutput = (float **) malloc(sizeof(float) * NoofCols);

/* Initializing The Output Matrices Elements As Zero */

for (i = 0; i < NoofCols; i++) {
	Checkoutput[i] = (float *) malloc(sizeof(float) * NoofRows);
	Trans[i] = (float *) malloc(sizeof(float) * NoofRows);
	for (j = 0; j < NoofRows; j++) {
		Checkoutput[i][j] = 0.0;
		Trans[i][j] = 0.0;
	}
}

gettimeofday(&TimeValue_Start, &TimeZone_Start);

/* OpenMP Parallel For Directive */
```

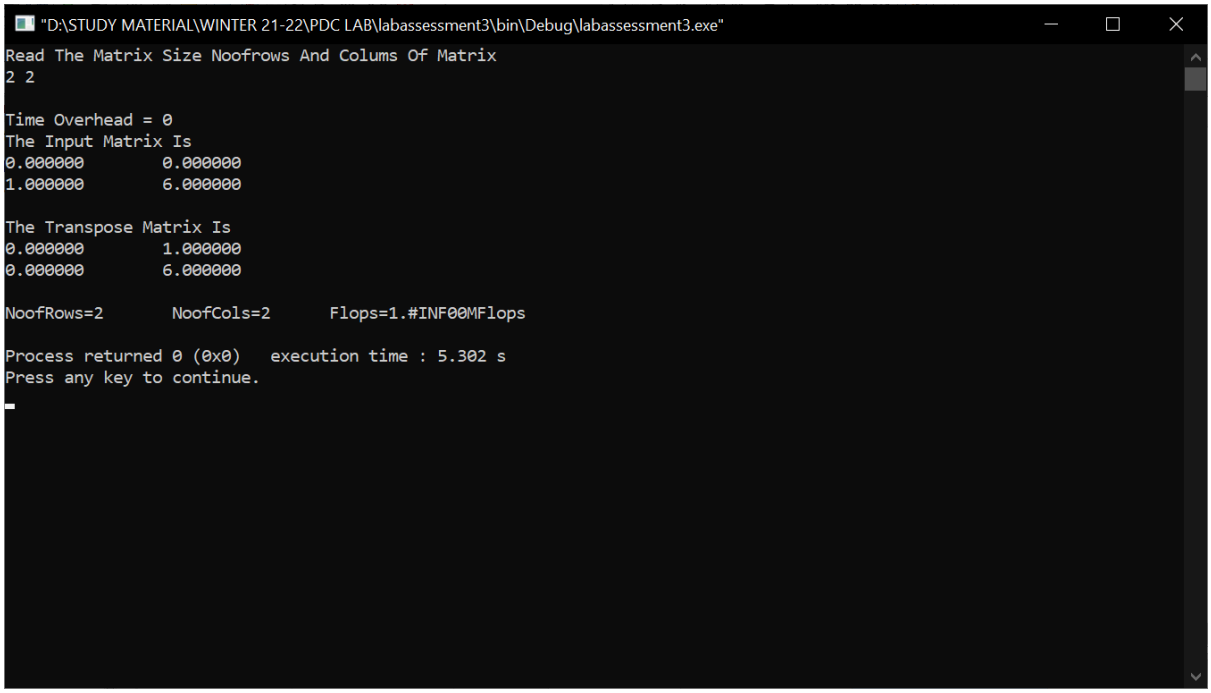
```
#pragma omp parallel for private(j)
        for (i = 0; i < NoofRows; i = i + 1)
                for (j = 0; j < NoofCols; j = j + 1)
                        Trans[j][i] = Matrix[i][j];

        gettimeofday(&TimeValue_Final, &TimeZone_Final);

        time_start = TimeValue_Start.tv_sec * 1000000 +
TimeValue_Start.tv_usec;
        time_end = TimeValue_Final.tv_sec * 1000000 +
TimeValue_Final.tv_usec;
        time_overhead = time_end - time_start;


        /* Serial Computation */

        for (i = 0; i < NoofRows; i = i + 1)
                for (j = 0; j < NoofCols; j = j + 1)
                        Checkoutput[j][i] = Matrix[i][j];

        for (i = 0; i < NoofCols; i = i + 1)
                for (j = 0; j < NoofRows; j = j + 1)
                        if (Checkoutput[i][j] == Trans[i][j])
                                continue;
                        else {
                                printf("There Is A Difference From Serial And
Parallel Calculation \n");
```

```
                exit(1);
            }

    printf("\nTime Overhead = %ld\n", time_overhead);

    printf("The Input Matrix Is \n");

    for (i = 0; i < NoofRows; i++) {
        for (j = 0; j < NoofCols; j++)
            printf("%f \t", Matrix[i][j]);
        printf("\n");
    }

    printf("\nThe Transpose Matrix Is \n");
    for (i = 0; i < NoofCols; i = i + 1) {
        for (j = 0; j < NoofRows; j = j + 1)
            printf("%f \t", Trans[i][j]);
        printf("\n");
    }

    /* Calculation Of Flops */

    flops = (float) 2 *NoofRows * NoofCols / (float) time_overhead;
    printf("\nNoofRows=%d\t NoofCols=%d \t Flops=%fMFlops\n",
NoofRows, NoofCols, flops);

    /* Freeing Allocated Memory */
```

```
    free(Matrix);
    free(Checkoutput);
    free(Trans);

}
```

## ***SCREENSHOT OF THE OUTPUT: -***



```
"D:\STUDY MATERIAL\WINTER 21-22\PDC LAB\labassessment3\bin\Debug\labassessment3.exe"
19BCE0660 NAKUL JADEJARead The Matrix Size Noofrows And Colums Of Matrix
2 2

Time Overhead = 0
The Input Matrix Is
0.000000        0.000000
1.000000        6.000000

The Transpose Matrix Is
0.000000        1.000000
0.000000        6.000000

NoofRows=2       NoofCols=2      Flops=1.#INF00MFlops

Process returned 0 (0x0)   execution time : 4.221 s
Press any key to continue.
```