

**Name : Abdelrahman shams eldin mohamed (40)**

## **OS LAB #2**

### **Threads**

#### **1) Matrix Multiplication :**

- **Major functions:**

1. The function procedure1() implements the first variation of the algorithm , it's basically creates individual thread for computation of every element to be exist in the result matrix by calling the function calculateElement() .
2. The function calculateElement() basically computes the element to be in indices i,k in the result array , it takes the row i from first matrix and the column k from the second matrix as inputs and outputs the computed element.
3. The function procedure2() implements the second variation of the algorithm by creating number of threads equal to number of rows in the result array and calling the function calculateRow() .
4. The function calculateRow() computes elements to be in the row i in the result matrix so it takes row i from first matrix as an input and outputs the row i to be in the result matrix.
5. The function initializeResultMatrix() is used to initialize the result matrix with zeroes so this function is called in main() after declaring the result array and before applying procedure 2.

- **Program flow:**

- ✓ First the two matrices to be multiplied are parsed from the input file then the result array is declared with rows equal to the rows of the first matrix and columns equal to the columns of the second matrix , then the result matrix is initialized by zeroes ,then procedure 1 is called and outputs its result matrix and elapsed time into the output file, then the result array is initialized again and procedure 2 is called ,then it outputs its result matrix and elapsed time into the output file.

- Sample runs and screen shots:

- Sample run #1:

Input file:



```
Open ▾ [icon] MMinput.txt ~/Desktop/OS_Lab_2 Save [menu] [min] [max] [close]

3 5
1 -2 3 4 5
1 2 -3 4 5
-1 2 3 4 5
5 4
-1 2 3 4
1 -2 3 4
1 2 -3 4
1 2 3 -4
-1 -2 -3 -4|

Plain Text ▾ Tab Width: 8 ▾ Ln 10, Col 12 ▾ INS
```

compiling and running of the program:

```
ab_shams@AB-Shams-HP:~/Desktop/OS_Lab_2$ gcc -pthread -o matMUL.o Matrix_Mul.c
ab_shams@AB-Shams-HP:~/Desktop/OS_Lab_2$ ./matMUL.o
Opening input file <MMinput.txt> .....

..... input file opened

parsing first matrix dimensions
n = 3
m = 5
parsing first Matrix :
      1      -2      3      4      5
      1       2     -3      4      5
     -1       2      3      4      5

parsing second matrix dimensions
m = 5
l = 4
```

parsing second Matrix :

-1	2	3	4
1	-2	3	4
1	2	-3	4
1	2	3	-4
-1	-2	-3	-4

input file <MMinput.txt> closed

initializing result Matrix with zeroes :

0	0	0	0
0	0	0	0
0	0	0	0

Method 1 started

opening output file <MMoutput.txt> .....

..... output file opened

now we are in threads .....

Thread #0 : C[0][0] = -1  
Thread #1 : C[0][1] = 10  
Thread #2 : C[0][2] = -15  
Thread #4 : C[1][0] = -3  
Thread #3 : C[0][3] = -28  
Thread #5 : C[1][1] = -10  
Thread #6 : C[1][2] = 15  
Thread #7 : C[1][3] = -36  
Thread #8 : C[2][0] = 5  
Thread #9 : C[2][1] = -2  
Thread #11 : C[2][3] = -20  
Thread #10 : C[2][2] = -9

..... threads ends now

elapsed time of procedure 1 in :

- seconds without ns: 0
- nanoseconds: 2406867
- total seconds: 2.406867e-03

Here is the Result Matrix:

-1	10	-15	-28
-3	-10	15	-36
5	-2	-9	-20

method 1 ended

initializing result Matrix with zeroes :

0      0      0      0

0      0      0      0

0      0      0      0

Method 2 started

now we are in threads .....

Thread #0 : working at row #0

Column #0 : C[0][0] = -1

Column #1 : C[0][1] = 10

Column #2 : C[0][2] = -15

Column #3 : C[0][3] = -28

Thread #2 : working at row #2

Column #0 : C[2][0] = 5

Column #1 : C[2][1] = -2

Column #2 : C[2][2] = -9

Column #3 : C[2][3] = -20

Thread #1 : working at row #1

Column #0 : C[1][0] = -3

Column #1 : C[1][1] = -10

Column #2 : C[1][2] = 15

Column #3 : C[1][3] = -36

..... threads ends now

elapsed time of procedure 2 in :

- seconds without ns: 0

- nanoseconds: 303485

- total seconds: 3.034850e-04

Here is the Result Matrix:

-1      10      -15      -28

-3      -10      15      -36

5      -2      -9      -20

method 2 ended

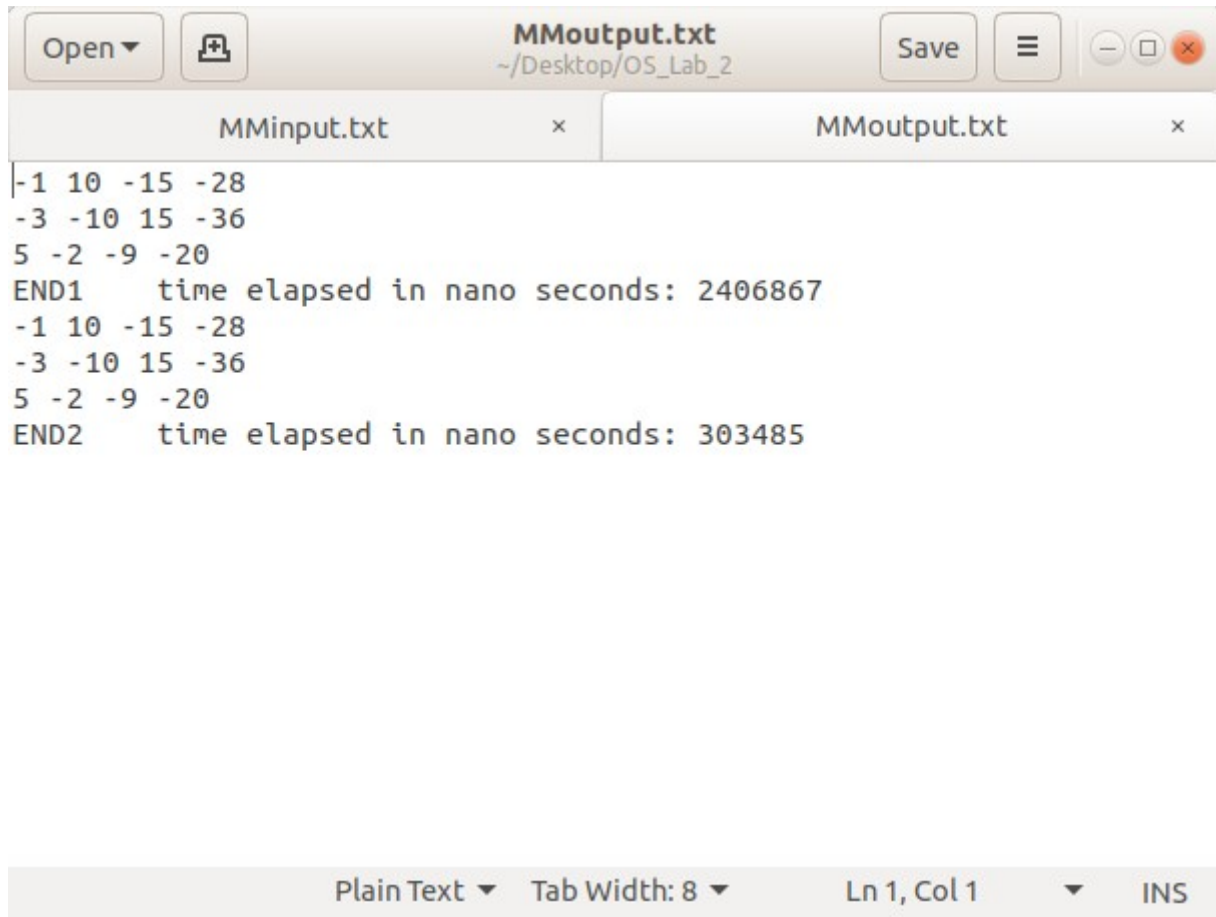
closing output file .....

output file <MMoutput> closed

\*\*\*\*\*program terminated successfully\*\*\*\*\*

ab\_shams@AB-Shams-HP:~/Desktop/OS\_Lab\_2\$

Output File:



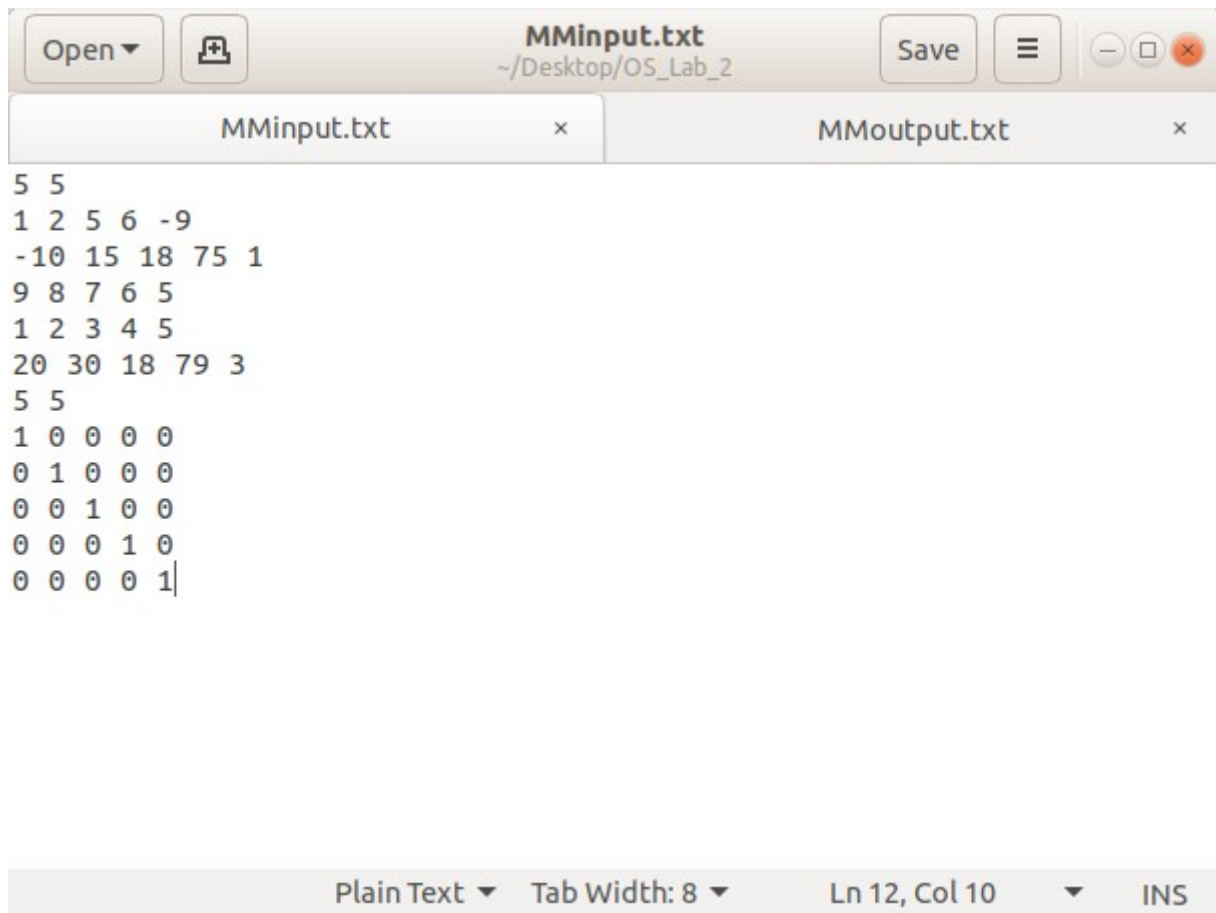
```
MMoutput.txt
~/Desktop/OS_Lab_2

MMinput.txt  x  MMoutput.txt  x
-1 10 -15 -28
-3 -10 15 -36
5 -2 -9 -20
END1    time elapsed in nano seconds: 2406867
-1 10 -15 -28
-3 -10 15 -36
5 -2 -9 -20
END2    time elapsed in nano seconds: 303485

Plain Text  Tab Width: 8  Ln 1, Col 1  INS
```

## ▪ Sample run #2:

Input file :



```
5 5
1 2 5 6 -9
-10 15 18 75 1
9 8 7 6 5
1 2 3 4 5
20 30 18 79 3
5 5
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
```

compiling and running program :

```
ab_shams@AB-Shams-HP:~/Desktop/OS_Lab_2$ gcc -pthread -o matMUL.o Matrix_Mul.c
ab_shams@AB-Shams-HP:~/Desktop/OS_Lab_2$ ./matMUL.o
Opening input file <MMinput.txt> .....
```

..... input file opened

parsing first matrix dimensions

n = 5

m = 5

parsing first Matrix :

1	2	5	6	-9
-10	15	18	75	1
9	8	7	6	5
1	2	3	4	5
20	30	18	79	3

parsing second matrix dimensions

m = 5

l = 5

parsing second Matrix :

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

input file <MMinput.txt> closed

initializing result Matrix with zeroes :

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Method 1 started

opening output file <MMoutput.txt> .....

..... output file opened

now we are in threads .....

Thread #1 : C[0][1] = 2

Thread #0 : C[0][0] = 1

Thread #2 : C[0][2] = 5

Thread #4 : C[0][4] = -9

Thread #3 : C[0][3] = 6

Thread #5 : C[1][0] = -10

Thread #6 : C[1][1] = 15

Thread #7 : C[1][2] = 18

Thread #8 : C[1][3] = 75

Thread #9 : C[1][4] = 1

Thread #10 : C[2][0] = 9

Thread #11 : C[2][1] = 8

Thread #12 : C[2][2] = 7

Thread #13 : C[2][3] = 6

Thread #14 : C[2][4] = 5

Thread #15 : C[3][0] = 1

Thread #16 : C[3][1] = 2  
Thread #17 : C[3][2] = 3  
Thread #18 : C[3][3] = 4  
Thread #19 : C[3][4] = 5  
Thread #20 : C[4][0] = 20  
Thread #21 : C[4][1] = 30  
Thread #22 : C[4][2] = 18  
Thread #23 : C[4][3] = 79  
Thread #24 : C[4][4] = 3

..... threads ends now

elapsed time of procedure 1 in :

- seconds without ns: 0
- nanoseconds: 806766
- total seconds: 8.067660e-04

Here is the Result Matrix:

1	2	5	6	-9
-10	15	18	75	1
9	8	7	6	5
1	2	3	4	5
20	30	18	79	3

method 1 ended

initializing result Matrix with zeroes :

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Method 2 started

now we are in threads .....

Thread #0 : working at row #0

Column #0 : C[0][0] = 1  
Column #1 : C[0][1] = 2  
Column #2 : C[0][2] = 5  
Column #3 : C[0][3] = 6  
Column #4 : C[0][4] = -9

Thread #1 : working at row #1

Column #0 : C[1][0] = -10



```

        Column #1 : C[1][1] = 15
        Column #2 : C[1][2] = 18
Thread #2 : working at row #2
Thread #3 : working at row #3
        Column #0 : C[2][0] = 9
        Column #3 : C[1][3] = 75
        Column #0 : C[3][0] = 1
        Column #1 : C[3][1] = 2
        Column #2 : C[3][2] = 3
        Column #3 : C[3][3] = 4
        Column #4 : C[3][4] = 5
        Column #1 : C[2][1] = 8
        Column #2 : C[2][2] = 7
        Column #3 : C[2][3] = 6
        Column #4 : C[2][4] = 5
Thread #4 : working at row #4
        Column #0 : C[4][0] = 20
        Column #1 : C[4][1] = 30
        Column #2 : C[4][2] = 18
        Column #3 : C[4][3] = 79
        Column #4 : C[4][4] = 3
        Column #4 : C[1][4] = 1

```

```

..... threads ends now
elapsed time of procedure 2 in :
    - seconds without ns: 0
    - nanoseconds: 367142
    - total seconds: 3.671420e-04

```

Here is the Result Matrix:

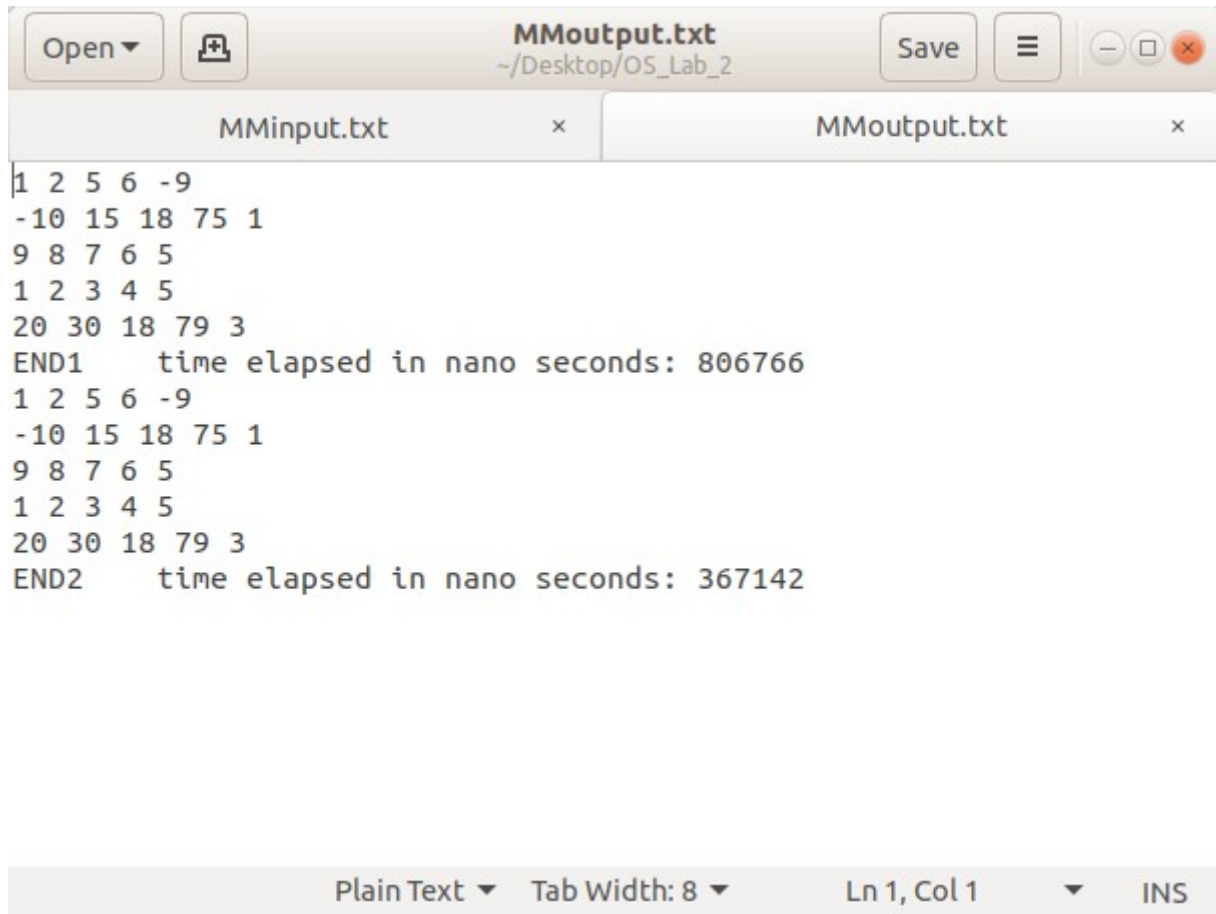
1	2	5	6	-9
-10	15	18	75	1
9	8	7	6	5
1	2	3	4	5
20	30	18	79	3

```

method 2 ended
closing output file .....
output file <MMoutput> closed
        *****program terminated successfully*****
ab_shams@AB-Shams-HP:~/Desktop/OS_Lab_2$

```

Output file:

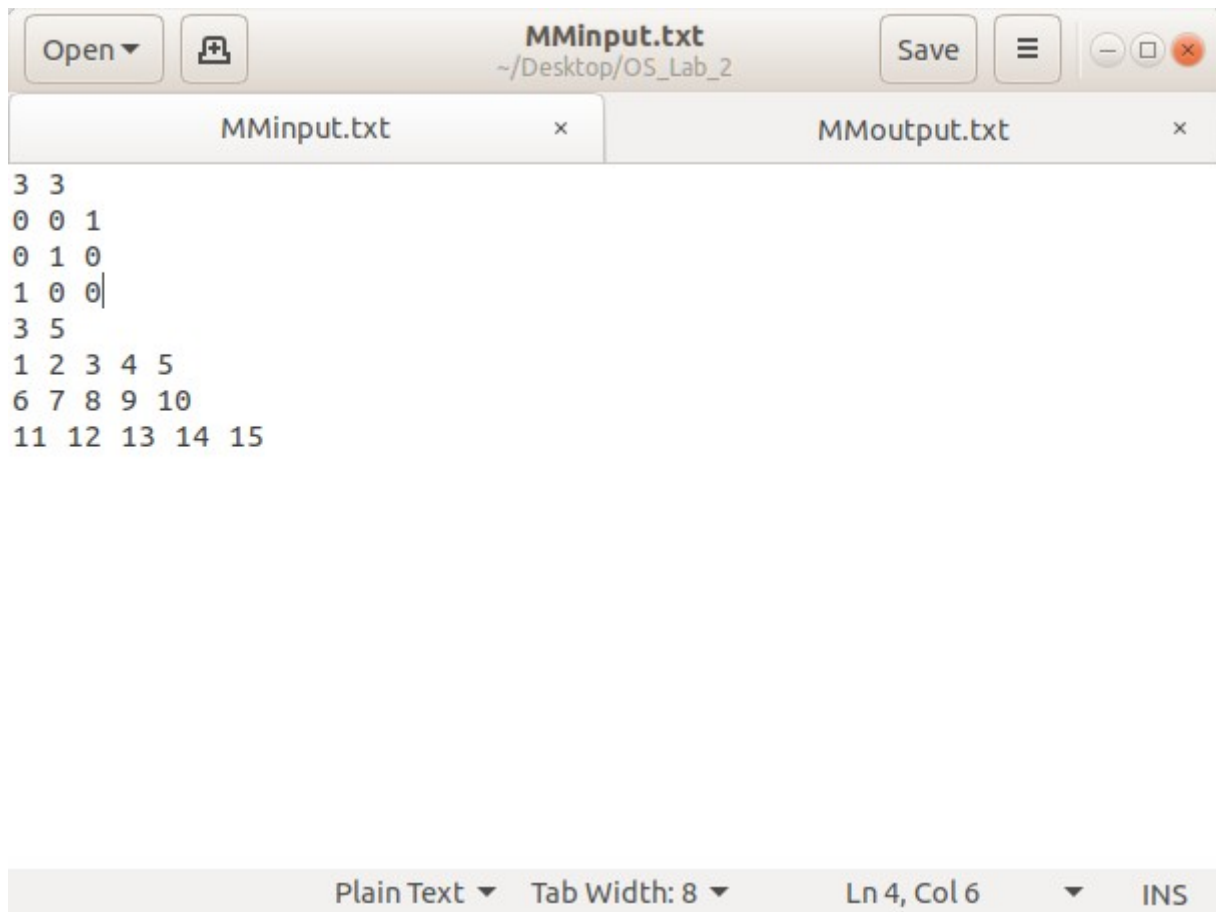


```
1 2 5 6 -9
-10 15 18 75 1
9 8 7 6 5
1 2 3 4 5
20 30 18 79 3
END1    time elapsed in nano seconds: 806766
1 2 5 6 -9
-10 15 18 75 1
9 8 7 6 5
1 2 3 4 5
20 30 18 79 3
END2    time elapsed in nano seconds: 367142
```

Plain Text ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾ INS

### ▪ Sample run #3:

Input file :



```
3 3
0 0 1
0 1 0
1 0 0
3 5
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
```

compiling an running program:

```
ab_shams@AB-Shams-HP:~/Desktop/OS_Lab_2$ gcc -pthread -o matMUL.o Matrix_Mul.c
ab_shams@AB-Shams-HP:~/Desktop/OS_Lab_2$ ./matMUL.o
Opening input file <MMinput.txt> .....

..... input file opened

parsing first matrix dimensions
n = 3
m = 3
parsing first Matrix :
      0      0      1
      0      1      0
      1      0      0

parsing second matrix dimensions
m = 3
l = 5
```

parsing second Matrix :

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

input file <MMinput.txt> closed

initializing result Matrix with zeroes :

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Method 1 started

opening output file <MMoutput.txt> .....

..... output file opened

now we are in threads .....

Thread #0 : C[0][0] = 11

Thread #1 : C[0][1] = 12

Thread #2 : C[0][2] = 13

Thread #3 : C[0][3] = 14

Thread #4 : C[0][4] = 15

Thread #5 : C[1][0] = 6

Thread #6 : C[1][1] = 7

Thread #7 : C[1][2] = 8

Thread #8 : C[1][3] = 9

Thread #9 : C[1][4] = 10

Thread #10 : C[2][0] = 1

Thread #11 : C[2][1] = 2

Thread #12 : C[2][2] = 3

Thread #13 : C[2][3] = 4

Thread #14 : C[2][4] = 5

..... threads ends now

elapsed time of procedure 1 in :

- seconds without ns: 0

- nanoseconds: 2460977

- total seconds: 2.460977e-03

Here is the Result Matrix:

11	12	13	14	15
6	7	8	9	10
1	2	3	4	5

method 1 ended

initializing result Matrix with zeroes :

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Method 2 started

now we are in threads .....

Thread #0 : working at row #0

Column #0 : C[0][0] = 11  
Column #1 : C[0][1] = 12  
Column #2 : C[0][2] = 13  
Column #3 : C[0][3] = 14  
Column #4 : C[0][4] = 15

Thread #1 : working at row #1

Column #0 : C[1][0] = 6  
Column #1 : C[1][1] = 7  
Column #2 : C[1][2] = 8  
Column #3 : C[1][3] = 9

Thread #2 : working at row #2

Column #0 : C[2][0] = 1  
Column #1 : C[2][1] = 2  
Column #2 : C[2][2] = 3  
Column #3 : C[2][3] = 4  
Column #4 : C[2][4] = 5  
Column #4 : C[1][4] = 10

..... threads ends now

elapsed time of procedure 2 in :

- seconds without ns: 0  
- nanoseconds: 500558  
- total seconds: 5.005580e-04

Here is the Result Matrix:

11	12	13	14	15
6	7	8	9	10
1	2	3	4	5

method 2 ended

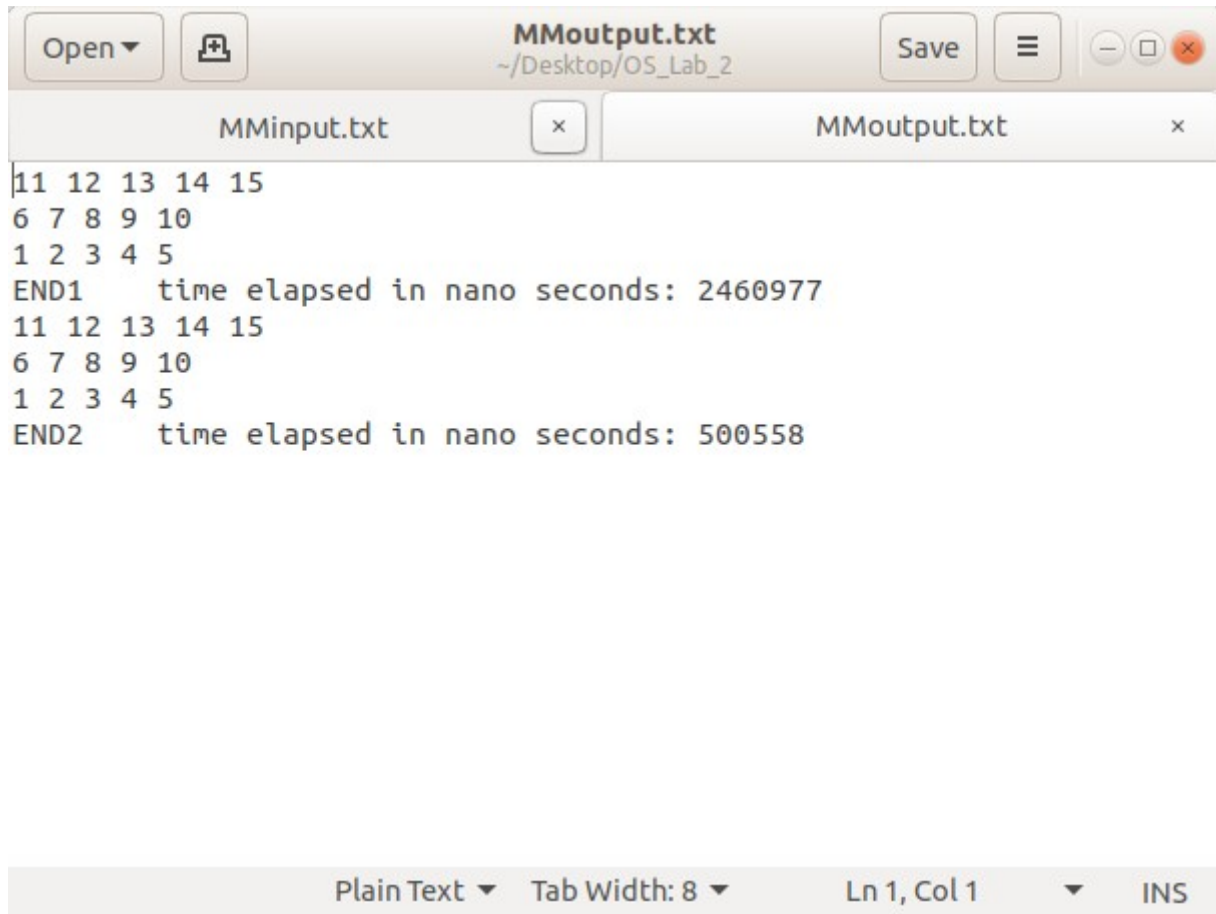
closing output file .....

output file <MMoutput> closed

\*\*\*\*\*program terminated successfully\*\*\*\*\*

ab\_shams@AB-Shams-HP:~/Desktop/OS\_Lab\_2\$

Output file:



```
11 12 13 14 15
6 7 8 9 10
1 2 3 4 5
END1    time elapsed in nano seconds: 2460977
11 12 13 14 15
6 7 8 9 10
1 2 3 4 5
END2    time elapsed in nano seconds: 500558
```

Plain Text ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾ INS

## 2) Merge Sort:

### ○ Major functions:

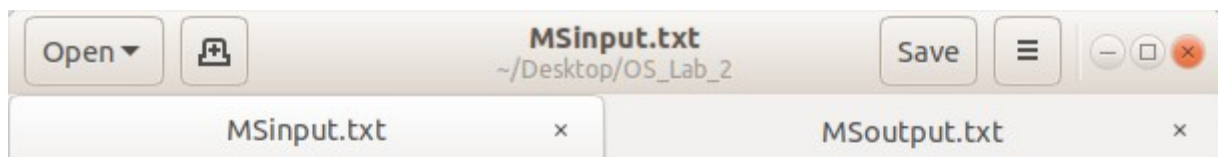
1. The function mergeSort() called by every thread created (initially we create a major thread in main for elements from  $i = 0$  to  $I = n - 1$ , so it sorts the whole array), this function works as basic mergeSort function (with out threads) but the difference here that we create individual thread for each sub-problems (i.e. the left sub-problem and right sub-problem after we divide the array into two arrays).
2. The function merge() merges two sorted array into one sorted array.

### ○ Program flow:

- ✓ First the array to be sorted is parsed from the input file then major thread are called (to sort the whole array), the thread function mergeSort then applies divide and conquer approach to sort the array, then the array after sorting and elapsed time are outputted into the output file.

### ○ Sample runs and screen shots:

Input file :



```
10
100 20 15 3 4 8 -7 -1 0 33|
```

compiling and running:

```
ab_shams@AB-Shams-HP:~/Desktop/OS_Lab_2$ gcc -pthread -o merge.o Merge_Sort.c
ab_shams@AB-Shams-HP:~/Desktop/OS_Lab_2$ ./merge.o
Opening input file <MSinput.txt> .....

..... input file opened

parsing array size .....
n = 10
Parsing array elements .....

Array before sorting : 100 20 15 3 4 8 -7 -1 0 33

input file <MSinput.txt> closed

major thread started.....

..... major thread ended

          ***** sorting done successfully *****

elapsed time in:
    - seconds without ns: 0
    - nanoseconds: 1677005
    - total seconds: 1.677005e-03
opening output file <MSoutput.txt> .....

..... output file opened

Array after sorting : -7 -1 0 3 4 8 15 20 33 100

output file <MSoutput> closed
ab_shams@AB-Shams-HP:~/Desktop/OS_Lab_2$
```



Output file:

