OS Lab 2 (Matrix Multi-threaded Multiplication)

Name: Abdelmoneim Hany Abdelmoneim Mohamed

ID: 19017359

1. Code Organization

I used some global variables for ease of reference, 9 functions apart from the main function.

Main calls SetUp which takes our command line arguments and checks them and reads from the files if no arguments are used then it uses default values.

SetUp calls readMatrix which reads a given matrix from file and assigns it.

Main calls validate which checks if the two arrays can be multiplied or not and allocates memory for resultant matrix.

Main calls multOneThread which creates one thread and passes MOT as a parameter to thread with no arguments and prints to relevant file.

Main calls multRowThread which creates threads with number of rows and passes MLT as a parameter to thread with rowIndex as argument and prints to relevant file.

Main calls multElementThread which creates threads with number of elements (rows x columns) and passes MET as a parameter to thread with elemDim struct as argument and prints to relevant file.

Then main frees all dynamically allocated matrices.

2. Main Functions

- SetUp
- readMatrix
- validate
- multOneThread
- multRowThread
- multELementThread

All of them explained in comments in code and in Code Organization.

3. **User Guide**

There is an Executable folder that has an executable and some text files. You can compile the main.c file directly by using this command:

gcc -pthread main.c -o matMultp

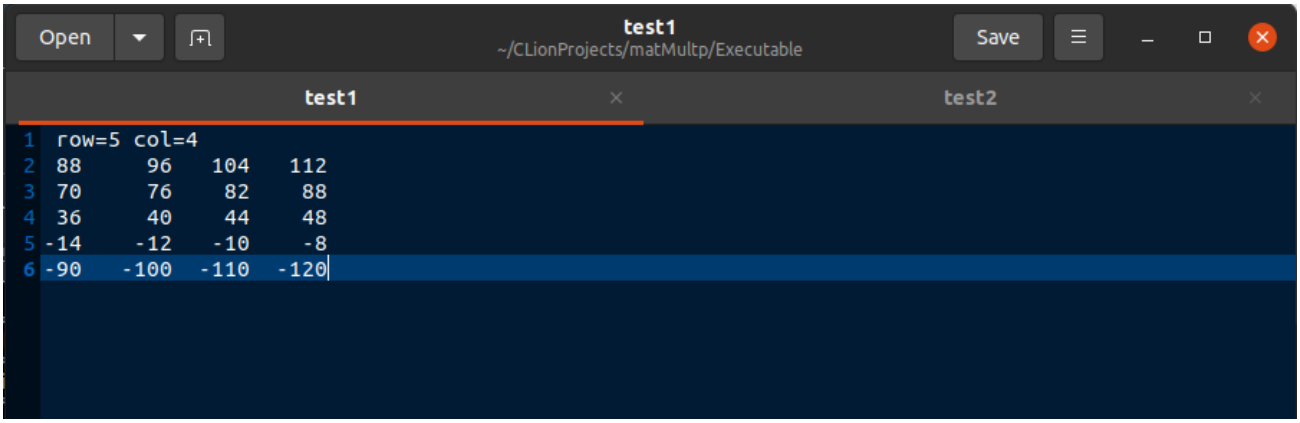
or by using the Makefile by opening bash in the directory containing main.c and Makefile and writing make in bash and a matMultp exectuable will be created.

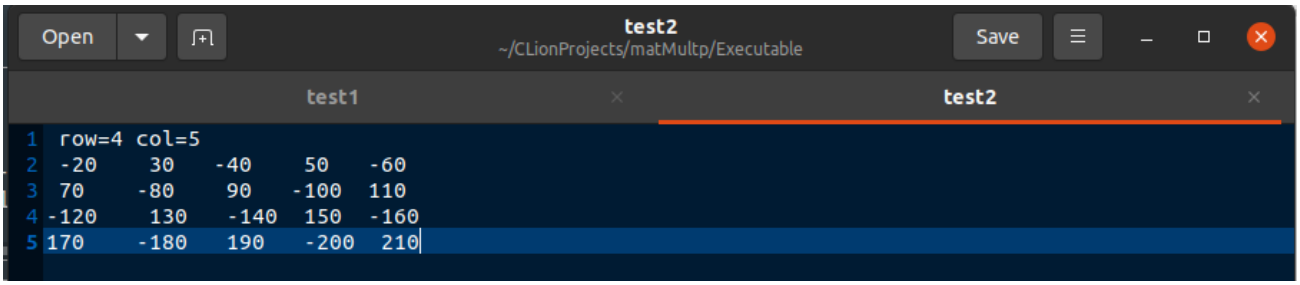
You can then use the exe to run the program whether with arguments or not.

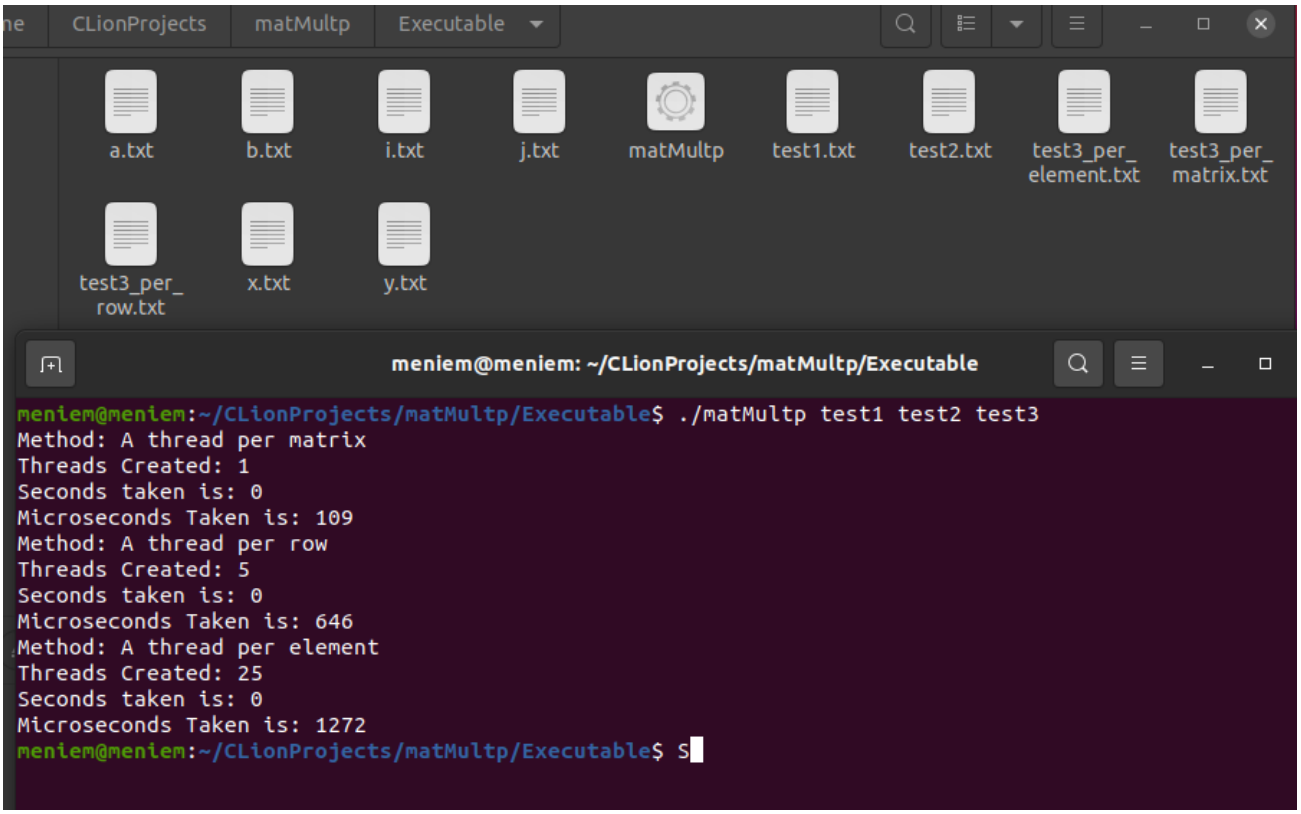
If you use no arguments then you can use ./matMultp directly in the right directory but you must make sure to have default txt files available like a.txt b.txt and 3 c txt files will be created each one with a different method.

If you use arguments then you can use ./matMultp arg1 arg2 outputarg3 in the right directly but you have to make sure that the name of both arg1 and arg2 is correct and there exist txt files with the same names in the directory, and 3 outputarg3 files will be created each one with a different method.

4. Sample runs







```
Method: A thread per matrix
row=5 col=5
11520.00 -11680.00 11840.00 -12000.00 12160.00
9040.00 -9160.00 9280.00 -9400.00 9520.00
4960.00 -5040.00 5120.00 -5200.00 5280.00
-720.00 680.00 -640.00 600.00 -560.00
-12400.00 12600.00 -12800.00 13000.00 -13200.00
```

```
Method: A thread per row
row=5 col=5
11520.00 -11680.00 11840.00 -12000.00 12160.00
9040.00 -9160.00 9280.00 -9400.00 9520.00
4960.00 -5040.00 5120.00 -5200.00 5280.00
-720.00 680.00 -640.00 600.00 -560.00
-12400.00 12600.00 -12800.00 13000.00 -13200.00
```

```
Method: A thread per Selement
row=5 col=5
11520.00 -11680.00 11840.00 -12000.00 12160.00
9040.00 -9160.00 9280.00 -9400.00 9520.00
4960.00 -5040.00 5120.00 -5200.00 5280.00
-720.00 680.00 -640.00 600.00 -560.00
-12400.00 12600.00 -12800.00 13000.00 -13200.00
```

Open ~/CLionProjects/matMultp/Executable **test4.txt** Save

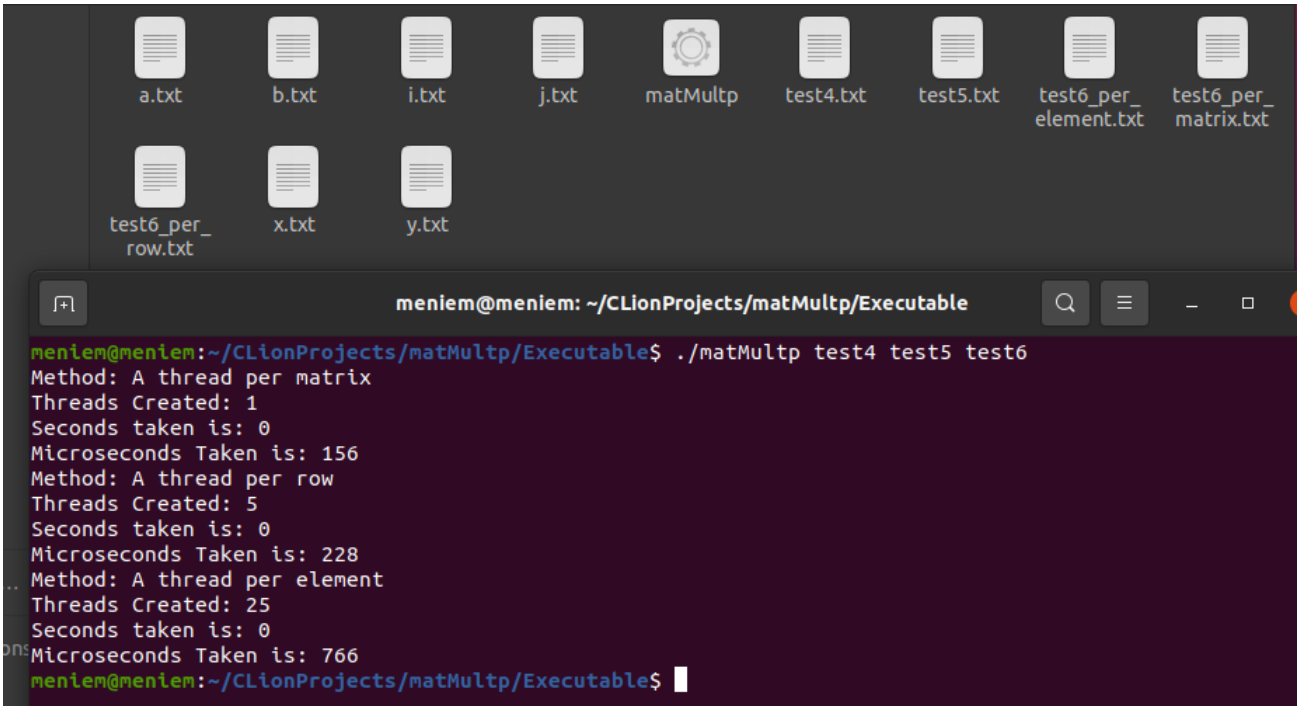
test4.txt | test5.txt

```
row=5 col=5
 11520      -11680       11840      -12000       12160
  9040       -9160        9280       -9400        9520
  4960       -5040        5120       -5200        5280
  -720         680        -640         600        -560
 -12400      12600      -12800       13000      -13200
```

Open ~/CLionProjects/matMultp/Executable **test5.txt** Save

test4.txt | test5.txt

```
row=5 col=5
215    230    245    260    275
490    530    570    610    650
765    830    895    960   1025
1040  1130   1220   1310   1400
1315  1430   1545   1660   1775
```



test6_per_matrix.txt
~/CLionProjects/matMultp/Executable

```
Method: A thread per matrix
row=5 col=5
9321600.00 10115200.00 10908800.00 11702400.00 12496000.00
7297200.00 7918400.00 8539600.00 9160800.00 9782000.00
4048800.00 4393600.00 4738400.00 5083200.00 5428000.00
-423600.00 -459200.00 -494800.00 -530400.00 -566000.00
-10122000.00 -10984000.00 -11846000.00 -12708000.00 -13570000.00
```

test6_per_row.txt
~/CLionProjects/matMultp/Executable

```
Method: A thread per row
row=5 col=5
9321600.00 10115200.00 10908800.00 11702400.00 12496000.00
7297200.00 7918400.00 8539600.00 9160800.00 9782000.00
4048800.00 4393600.00 4738400.00 5083200.00 5428000.00
-423600.00 -459200.00 -494800.00 -530400.00 -566000.00
-10122000.00 -10984000.00 -11846000.00 -12708000.00 -13570000.00
```

*test6_per_element.txt
~/CLionProjects/matMultp/Executable

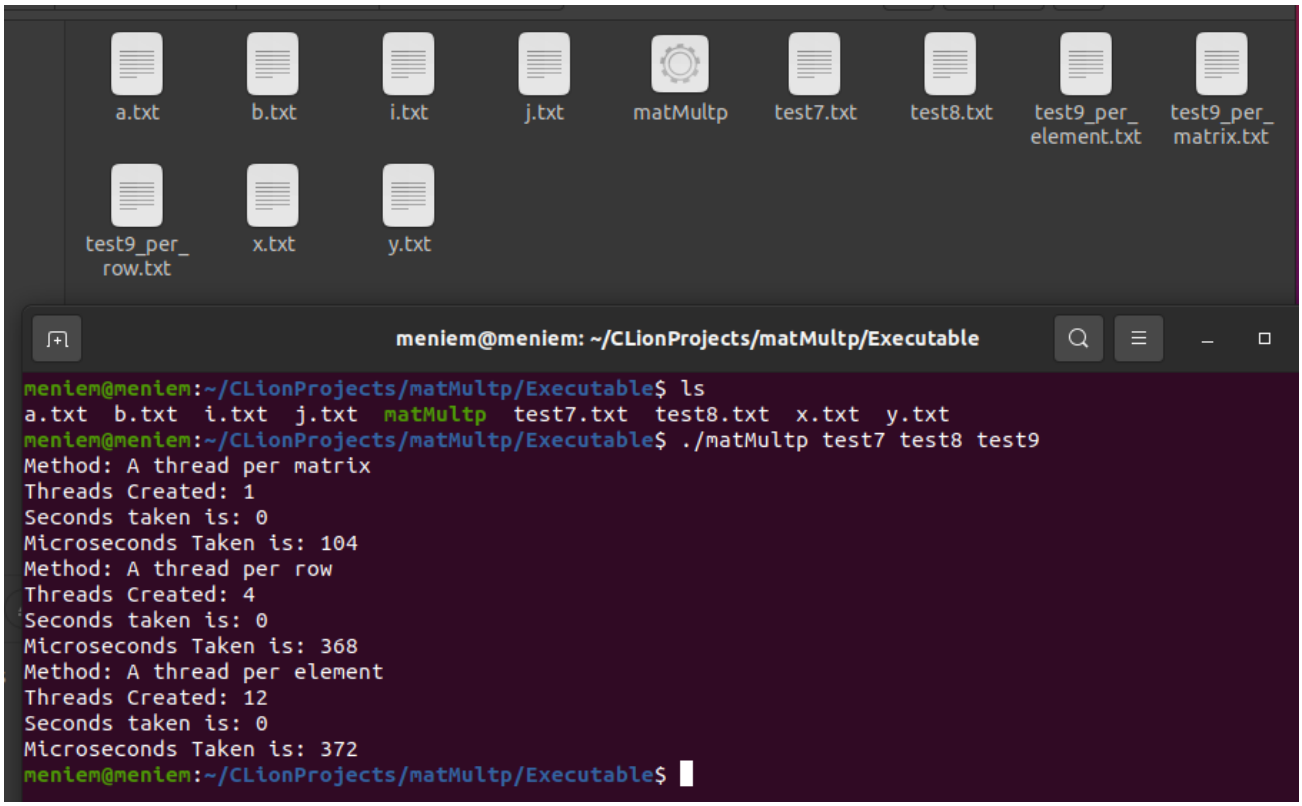
```
Method: A thread per element
row=5 col=5
9321600.00 10115200.00 10908800.00 11702400.00 12496000.00
7297200.00 7918400.00 8539600.00 9160800.00 9782000.00
4048800.00 4393600.00 4738400.00 5083200.00 5428000.00
-423600.00 -459200.00 -494800.00 -530400.00 -566000.00
-10122000.00 -10984000.00 -11846000.00 -12708000.00 -13570000.00
```

test7

```
row=4 col=6
1 2 3 4 5 6
-7 -8 -9 -10 -11 -12
12 13 14 15 16 17
-18 -19 -20 -21 -22 -23
```

test8

```
row=6 col=3
100 99 98
-97 -96 -95
94 93 92
-91 -90 -89
88 87 86
-85 -84 -83
```



```
Method: A thread per matrix
row=4 col=3
-246.00 -243.00 -240.00
192.00 189.00 186.00
-147.00 -144.00 -141.00
93.00 90.00 87.00
```

```
Method: A thread per row
row=4 col=3
-246.00 -243.00 -240.00
192.00 189.00 186.00
-147.00 -144.00 -141.00
93.00 90.00 87.00
```

```
Method: A thread per element
row=4 col=3
-246.00 -243.00 -240.00
192.00 189.00 186.00
-147.00 -144.00 -141.00
93.00 90.00 87.00
```

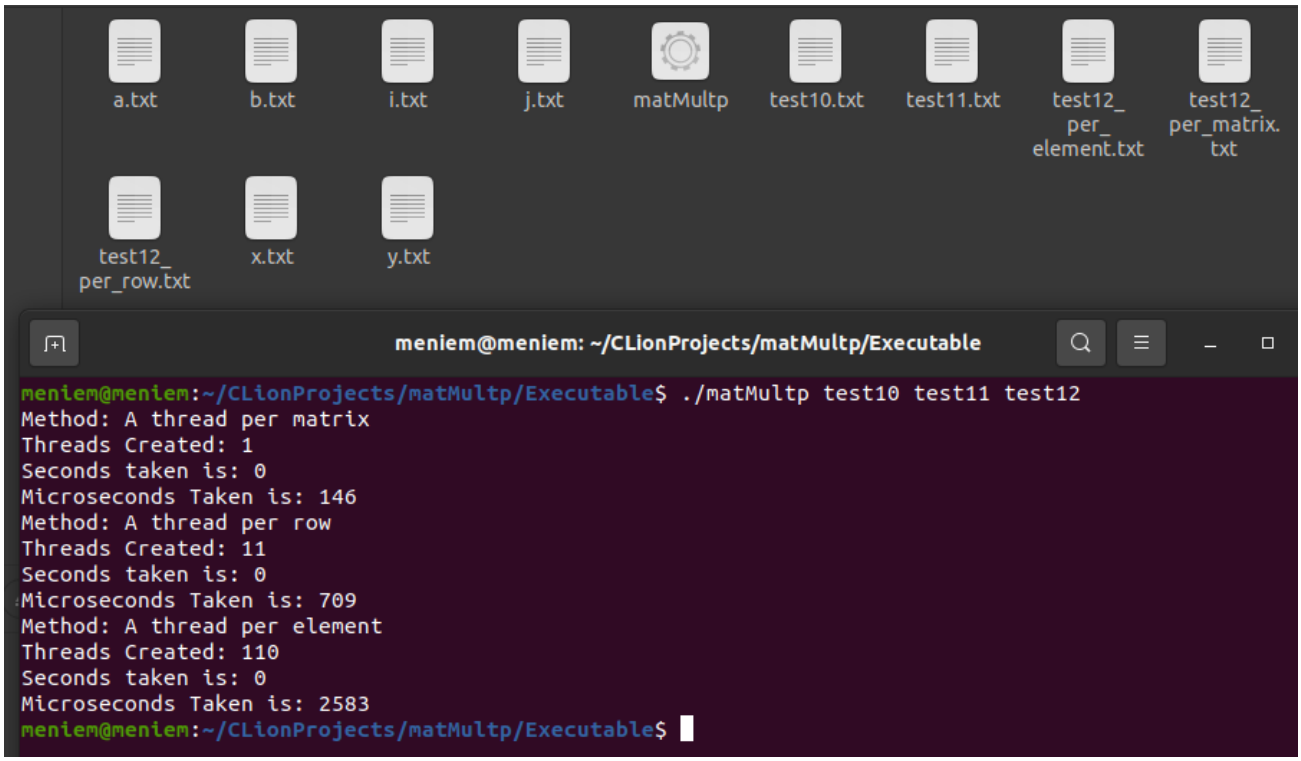
test10.txt ~/CLionProjects...

```
row=11 col=3
11 22 33
44 55 66
77 88 77
-88 -99 -111
-222 -333 -444
-555 0 12
21 23 32
34 43 45
-54 -56 -65
-67 -76 -78
-87 -89 98
```

Plain Text | Tab Width: 6 | Ln 9, Col 11 | INS

test11.txt ~/CLionProjects/matM...

```
row=3 col=10
1 2 3 4 5 6 7 8 9 10
-11 -12 -13 -14 -15 -16 -17 -18 -19 -20
21 22 23 24 25 26 27 28 29 30
```



test12_per_matrix.txt
~/CLionProjects/matMultp/Executable

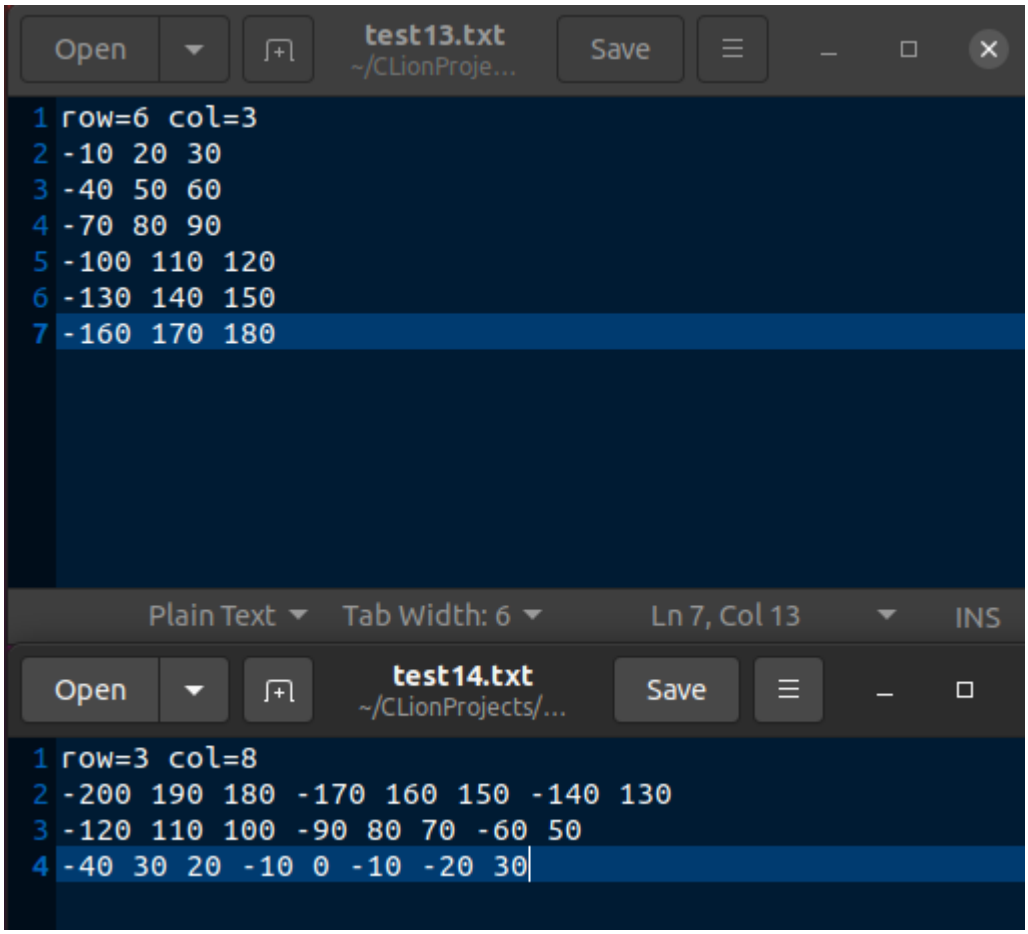
```
Method: A thread per matrix
row=11 col=10
462.00 484.00 506.00 528.00 550.00 572.00 594.00 616.00 638.00 660.00
825.00 880.00 935.00 990.00 1045.00 1100.00 1155.00 1210.00 1265.00 1320.00
726.00 792.00 858.00 924.00 990.00 1056.00 1122.00 1188.00 1254.00 1320.00
-1330.00 -1430.00 -1530.00 -1630.00 -1730.00 -1830.00 -1930.00 -2030.00 -2130.00 -2230.00
-5883.00 -6216.00 -6549.00 -6882.00 -7215.00 -7548.00 -7881.00 -8214.00 -8547.00 -8880.00
-303.00 -846.00 -1389.00 -1932.00 -2475.00 -3018.00 -3561.00 -4104.00 -4647.00 -5190.00
440.00 470.00 500.00 530.00 560.00 590.00 620.00 650.00 680.00 710.00
506.00 542.00 578.00 614.00 650.00 686.00 722.00 758.00 794.00 830.00
-803.00 -866.00 -929.00 -992.00 -1055.00 -1118.00 -1181.00 -1244.00 -1307.00 -1370.00
-869.00 -938.00 -1007.00 -1076.00 -1145.00 -1214.00 -1283.00 -1352.00 -1421.00 -1490.00
2950.00 3050.00 3150.00 3250.00 3350.00 3450.00 3550.00 3650.00 3750.00 3850.00
```

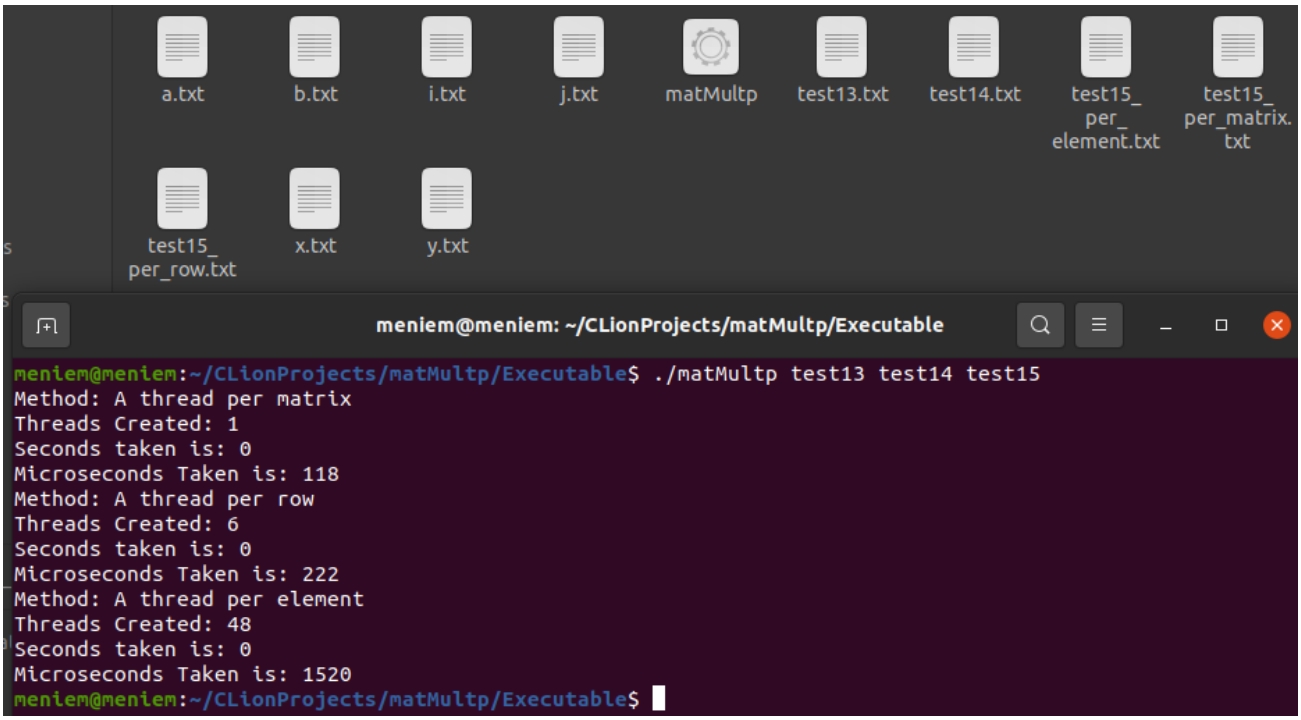
test12_per_row.txt
~/CLionProjects/matMultp/Executable

```
Method: A thread per row
row=11 col=10
462.00 484.00 506.00 528.00 550.00 572.00 594.00 616.00 638.00 660.00
825.00 880.00 935.00 990.00 1045.00 1100.00 1155.00 1210.00 1265.00 1320.00
726.00 792.00 858.00 924.00 990.00 1056.00 1122.00 1188.00 1254.00 1320.00
-1330.00 -1430.00 -1530.00 -1630.00 -1730.00 -1830.00 -1930.00 -2030.00 -2130.00 -2230.00
-5883.00 -6216.00 -6549.00 -6882.00 -7215.00 -7548.00 -7881.00 -8214.00 -8547.00 -8880.00
-303.00 -846.00 -1389.00 -1932.00 -2475.00 -3018.00 -3561.00 -4104.00 -4647.00 -5190.00
440.00 470.00 500.00 530.00 560.00 590.00 620.00 650.00 680.00 710.00
506.00 542.00 578.00 614.00 650.00 686.00 722.00 758.00 794.00 830.00
-803.00 -866.00 -929.00 -992.00 -1055.00 -1118.00 -1181.00 -1244.00 -1307.00 -1370.00
-869.00 -938.00 -1007.00 -1076.00 -1145.00 -1214.00 -1283.00 -1352.00 -1421.00 -1490.00
2950.00 3050.00 3150.00 3250.00 3350.00 3450.00 3550.00 3650.00 3750.00 3850.00
```

test12_per_element.txt
~/CLionProjects/matMultp/Executable

```
Method: A thread per element
row=11 col=10
462.00 484.00 506.00 528.00 550.00 572.00 594.00 616.00 638.00 660.00
825.00 880.00 935.00 990.00 1045.00 1100.00 1155.00 1210.00 1265.00 1320.00
726.00 792.00 858.00 924.00 990.00 1056.00 1122.00 1188.00 1254.00 1320.00
-1330.00 -1430.00 -1530.00 -1630.00 -1730.00 -1830.00 -1930.00 -2030.00 -2130.00 -2230.00
-5883.00 -6216.00 -6549.00 -6882.00 -7215.00 -7548.00 -7881.00 -8214.00 -8547.00 -8880.00
-303.00 -846.00 -1389.00 -1932.00 -2475.00 -3018.00 -3561.00 -4104.00 -4647.00 -5190.00
440.00 470.00 500.00 530.00 560.00 590.00 620.00 650.00 680.00 710.00
506.00 542.00 578.00 614.00 650.00 686.00 722.00 758.00 794.00 830.00
-803.00 -866.00 -929.00 -992.00 -1055.00 -1118.00 -1181.00 -1244.00 -1307.00 -1370.00
-869.00 -938.00 -1007.00 -1076.00 -1145.00 -1214.00 -1283.00 -1352.00 -1421.00 -1490.00
2950.00 3050.00 3150.00 3250.00 3350.00 3450.00 3550.00 3650.00 3750.00 3850.00
```


Open
test13.txt
~/CLionProje...
Save
row=6 col=3
-10 20 30
-40 50 60
-70 80 90
-100 110 120
-130 140 150
-160 170 180
Plain Text
Tab Width: 6
Ln 7, Col 13
INS
Open
test14.txt
~/CLionProjects/...
Save
row=3 col=8
-200 190 180 -170 160 150 -140 130
-120 110 100 -90 80 70 -60 50
-40 30 20 -10 0 -10 -20 30


a.txt
b.txt
i.txt
j.txt
matMultp
test13.txt
test14.txt
test15_per_element.txt
test15_per_matrix.txt
test15_per_row.txt
x.txt
y.txt
meniem@meniem: ~/CLionProjects/matMultp/Executable
meniem@meniem:~/CLionProjects/matMultp/Executable$ ./matMultp test13 test14 test15
Method: A thread per matrix
Threads Created: 1
Seconds taken is: 0
Microseconds Taken is: 118
Method: A thread per row
Threads Created: 6
Seconds taken is: 0
Microseconds Taken is: 222
Method: A thread per element
Threads Created: 48
Seconds taken is: 0
Microseconds Taken is: 1520
meniem@meniem:~/CLionProjects/matMultp/Executable$

test15_per_matrix.txt
~/CLionProjects/matMultp/Executable

```
Method: A thread per matrix
row=6 col=8
-1600.00 1200.00 800.00 -400.00 0.00 -400.00 -400.00 600.00
-400.00 -300.00 -1000.00 1700.00 -2400.00 -3100.00 1400.00 -900.00
800.00 -1800.00 -2800.00 3800.00 -4800.00 -5800.00 3200.00 -2400.00
2000.00 -3300.00 -4600.00 5900.00 -7200.00 -8500.00 5000.00 -3900.00
3200.00 -4800.00 -6400.00 8000.00 -9600.00 -11200.00 6800.00 -5400.00
4400.00 -6300.00 -8200.00 10100.00 -12000.00 -13900.00 8600.00 -6900.00
```

test15_per_row.txt
~/CLionProjects/matMultp/Executable

```
Method: A thread per row
row=6 col=8
-1600.00 1200.00 800.00 -400.00 0.00 -400.00 -400.00 600.00
-400.00 -300.00 -1000.00 1700.00 -2400.00 -3100.00 1400.00 -900.00
800.00 -1800.00 -2800.00 3800.00 -4800.00 -5800.00 3200.00 -2400.00
2000.00 -3300.00 -4600.00 5900.00 -7200.00 -8500.00 5000.00 -3900.00
3200.00 -4800.00 -6400.00 8000.00 -9600.00 -11200.00 6800.00 -5400.00
4400.00 -6300.00 -8200.00 10100.00 -12000.00 -13900.00 8600.00 -6900.00
```

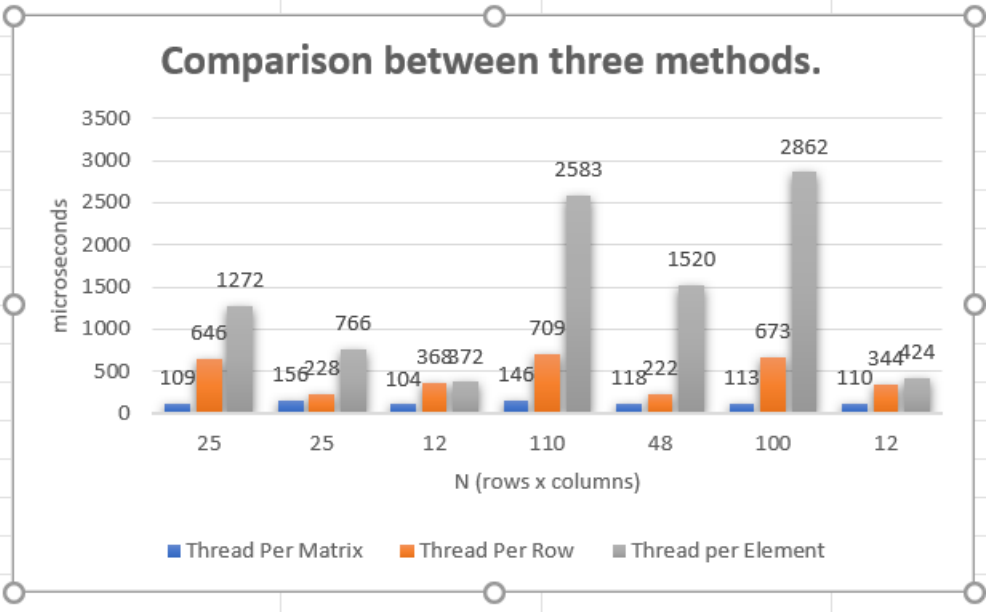
test15_per_element.txt
~/CLionProjects/matMultp/Executable

```
Method: A thread per element
row=6 col=8
-1600.00 1200.00 800.00 -400.00 0.00 -400.00 -400.00 600.00
-400.00 -300.00 -1000.00 1700.00 -2400.00 -3100.00 1400.00 -900.00
800.00 -1800.00 -2800.00 3800.00 -4800.00 -5800.00 3200.00 -2400.00
2000.00 -3300.00 -4600.00 5900.00 -7200.00 -8500.00 5000.00 -3900.00
3200.00 -4800.00 -6400.00 8000.00 -9600.00 -11200.00 6800.00 -5400.00
4400.00 -6300.00 -8200.00 10100.00 -12000.00 -13900.00 8600.00 -6900.00
```

Plain Text Tab Width: 6 Ln 1, Col 1 INS

5. Comparison of three methods

Thread Per Matrix	Thread Per Row	Thread per Element	N
109	646	1272	25
156	228	766	25
104	368	372	12
146	709	2583	110
118	222	1520	48
113	673	2862	100
110	344	424	12



As seen above using more threads leads to more time in our situation since our matrices are relatively small and the time of creation of one thread is a lot more costly than a couple of iterations on the matrix multiplication algorithm