Name: Bishoy Nader Fathy
ID: 22

Lab Assignment 2: Threads

21st October 2017

OVERVIEW

You are required to implement a multi-threaded matrix multiplication program. The input to the program is two matrixes A(x*y) and B(y*z) that are read from corresponding files. The output is a matrix C(x*z) that is written to an output file. A parallelized version of matrix multiplication can be done using one of these two methods: (1) a thread computes each row in the output C matrix, or (2) a thread computes each element in the output C matrix.

How the code is organized

The project is divided to 4 modules:

1. Main: the entry point to the project.
2. File Manager: handles reading from files and writing to files.
3. Environment: containing the input and output matrices data.
4. Multiplier: handles multiplication in both row and element methods.

Main Functions

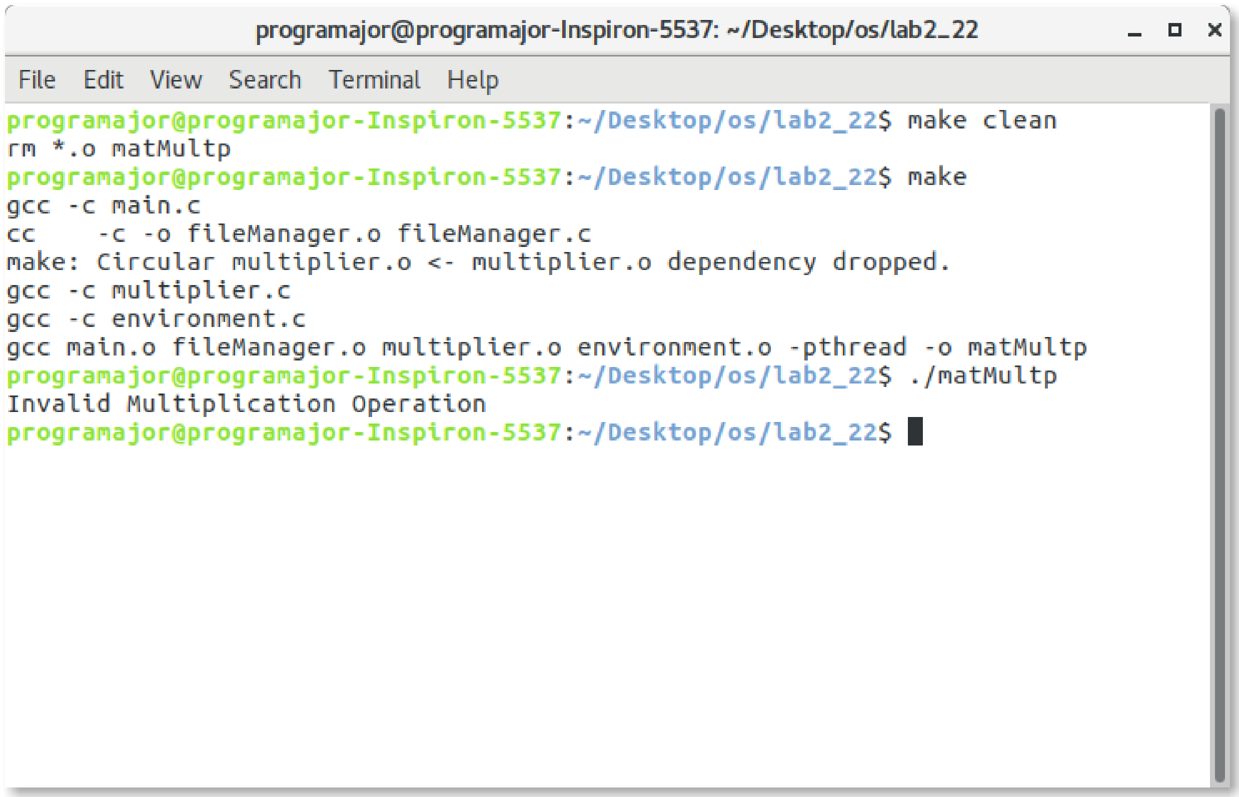
```
/**
* entry point to the application
* @param argc the number of arguments when running the app.
* @param argv the arguments when running the app.
*/
int main (int argc, char *argv[]);
```

```
/**
* Set Files with default values.
*/
void setDefaultFiles();
```

```
/**
* set the files for input and output
* @param first first input file.
* @param second second input file
* @param out output file.
*/
void setFiles(char *first, char *second, char *out);

/**
* reads the input matrices either from default files or selected files
*/
void readInput();

/**
* write the output matrix from row method to the file.
*/
void writeFirstOutput();

/**
* write the output matrix from element method to the file.
*/
void writeSecondOutput();

/**
* Multiplies both matrices by row method and element method.
*/
void startMultiplication();

/**
* multiply the two matrices with a thread for every row.
*/
void multiplyRowWise();

/**
* multiply the two matrices with a thread for every element.
*/
void multiplyElementWise();

/**
* executes on a thread responsible for a row in the output matrix.
* @param rowData contains the index of the row.
*/
void* calculateRow(void* rowData);
```

```
/**
* executes on a thread responsible for an element in the output matrix.
* @param elementData contains the index of the row and index of the column of the element.
*/
void* calculateElement(void* elementData);
```

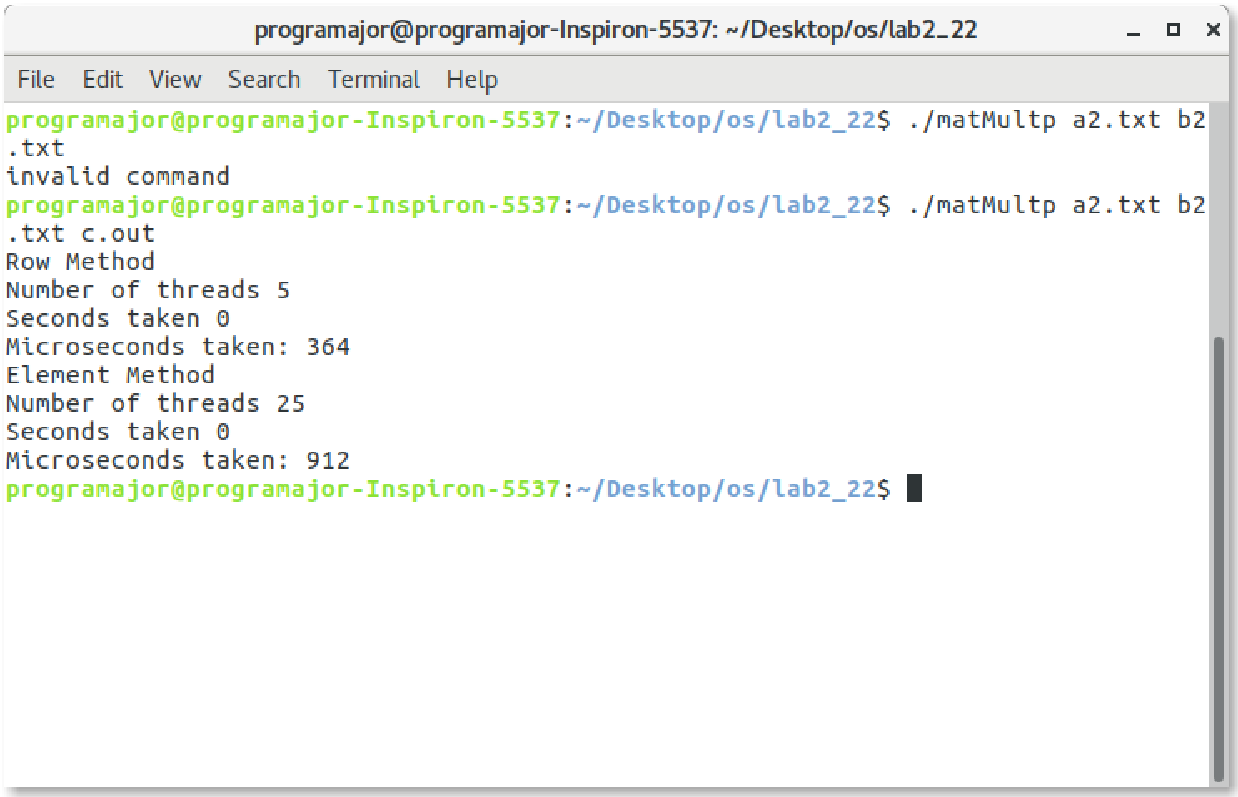
Sample Runs



programajor@programajor-Inspiron-5537: ~/Desktop/os/lab2_22
File Edit View Search Terminal Help

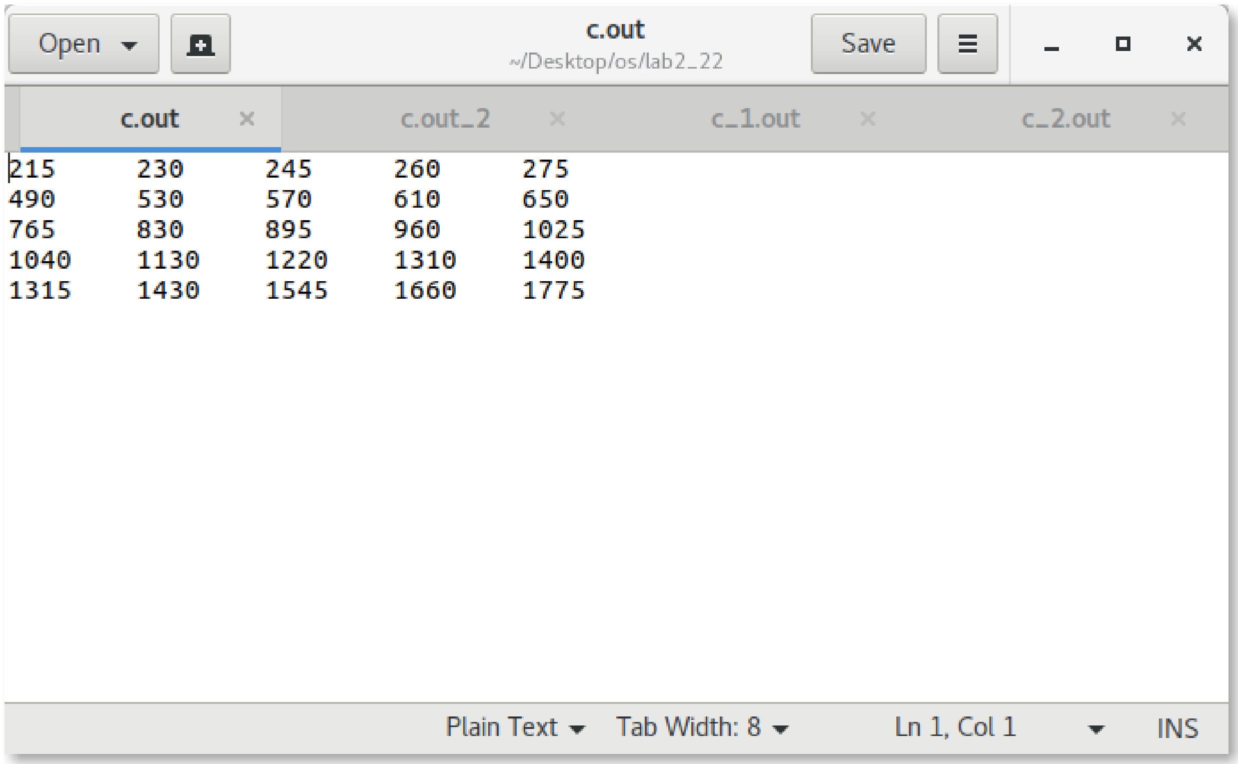
```
programajor@programajor-Inspiron-5537:~/Desktop/os/lab2_22$ make clean
rm *.o matMultp
programajor@programajor-Inspiron-5537:~/Desktop/os/lab2_22$ make
gcc -c main.c
cc    -c -o fileManager.o fileManager.c
make: Circular multiplier.o <- multiplier.o dependency dropped.
gcc -c multiplier.c
gcc -c environment.c
gcc main.o fileManager.o multiplier.o environment.o -pthread -o matMultp
programajor@programajor-Inspiron-5537:~/Desktop/os/lab2_22$ ./matMultp
Invalid Multiplication Operation
programajor@programajor-Inspiron-5537:~/Desktop/os/lab2_22$ 
```

Running application with a (5x5) matrix and a (4x4) matrix

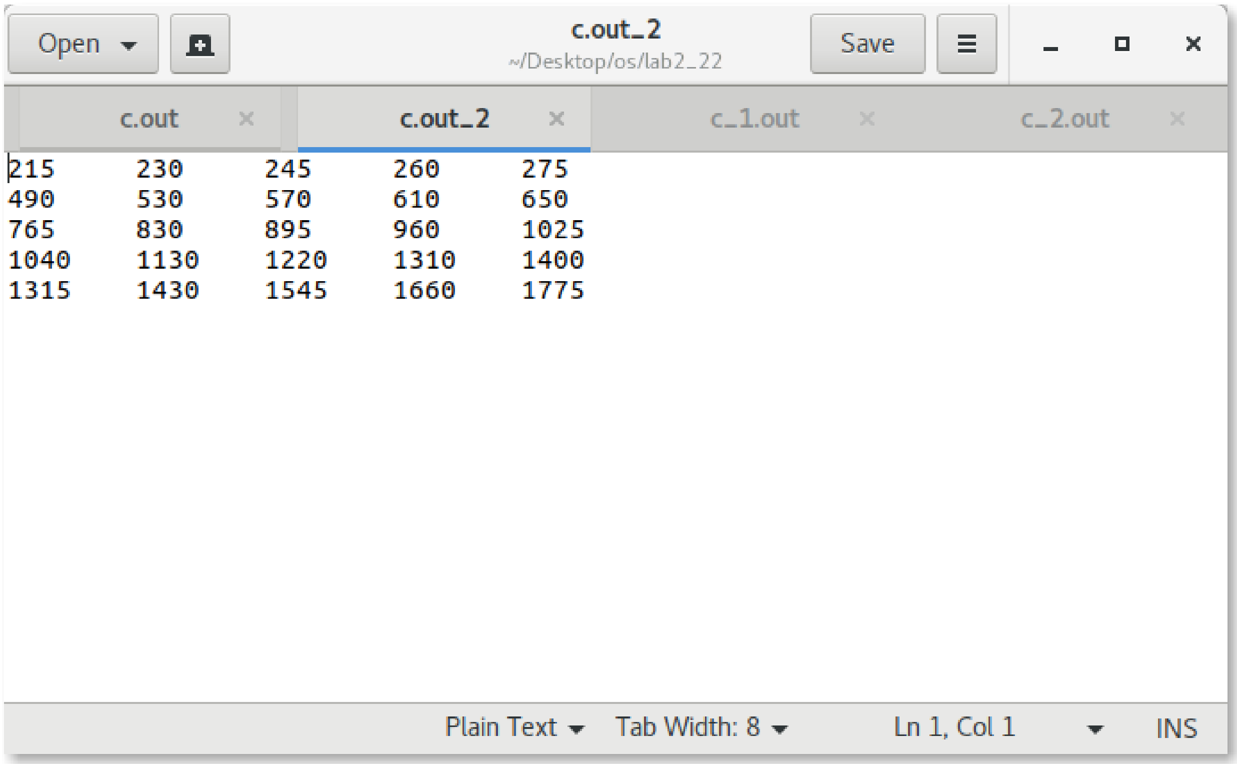

programajor@programajor-Inspiron-5537: ~/Desktop/os/lab2_22
File Edit View Search Terminal Help

```
programajor@programajor-Inspiron-5537:~/Desktop/os/lab2_22$ ./matMultp a2.txt b2
.txt
invalid command
programajor@programajor-Inspiron-5537:~/Desktop/os/lab2_22$ ./matMultp a2.txt b2
.txt c.out
Row Method
Number of threads 5
Seconds taken 0
Microseconds taken: 364
Element Method
Number of threads 25
Seconds taken 0
Microseconds taken: 912
programajor@programajor-Inspiron-5537:~/Desktop/os/lab2_22$
```

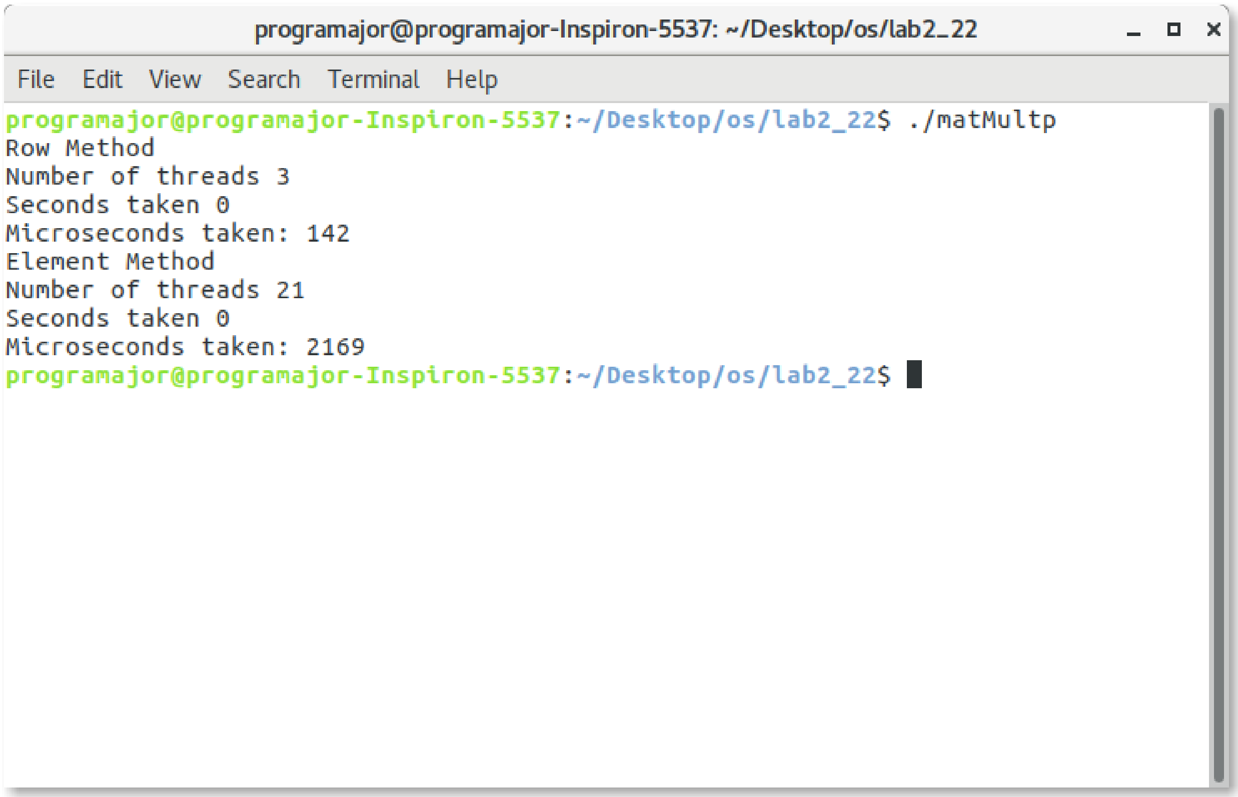
Running application with 2 (5x5) matrices from custom files



```
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
```

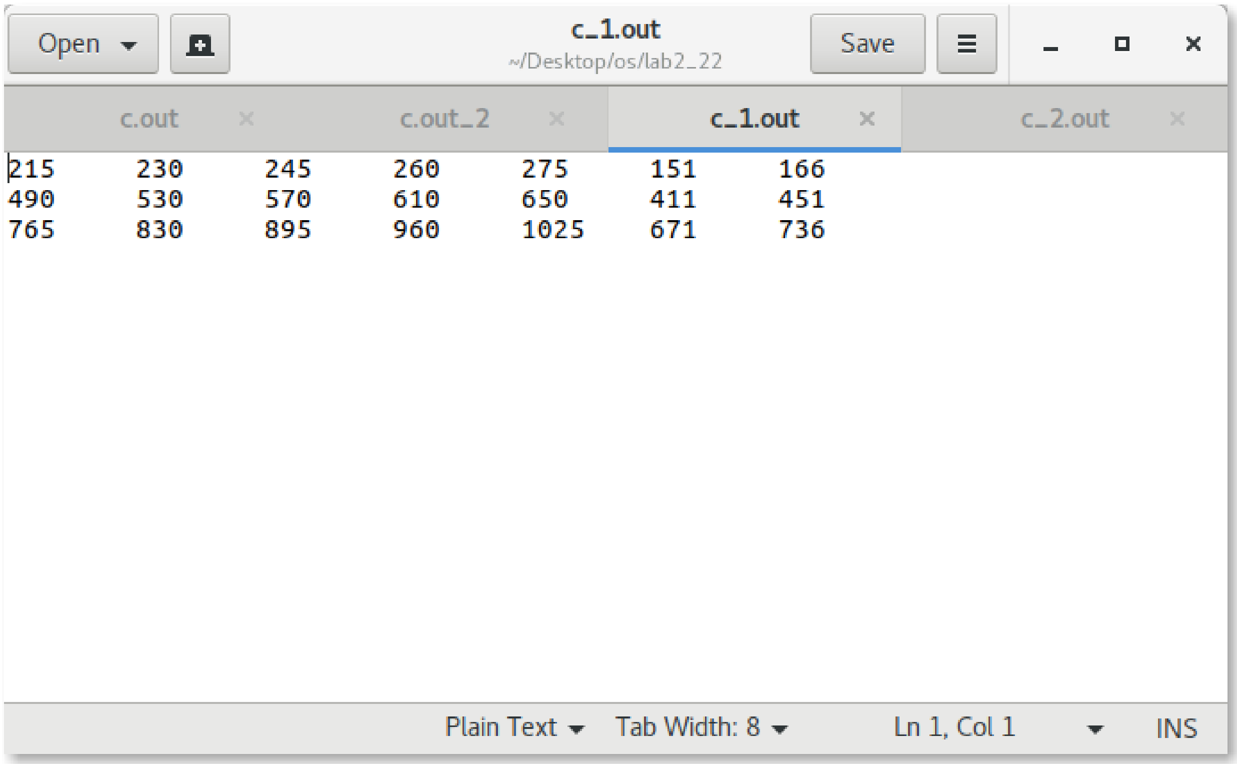


```
215	230	245	260	275
490	530	570	610	650
765	830	895	960	1025
1040	1130	1220	1310	1400
1315	1430	1545	1660	1775
```

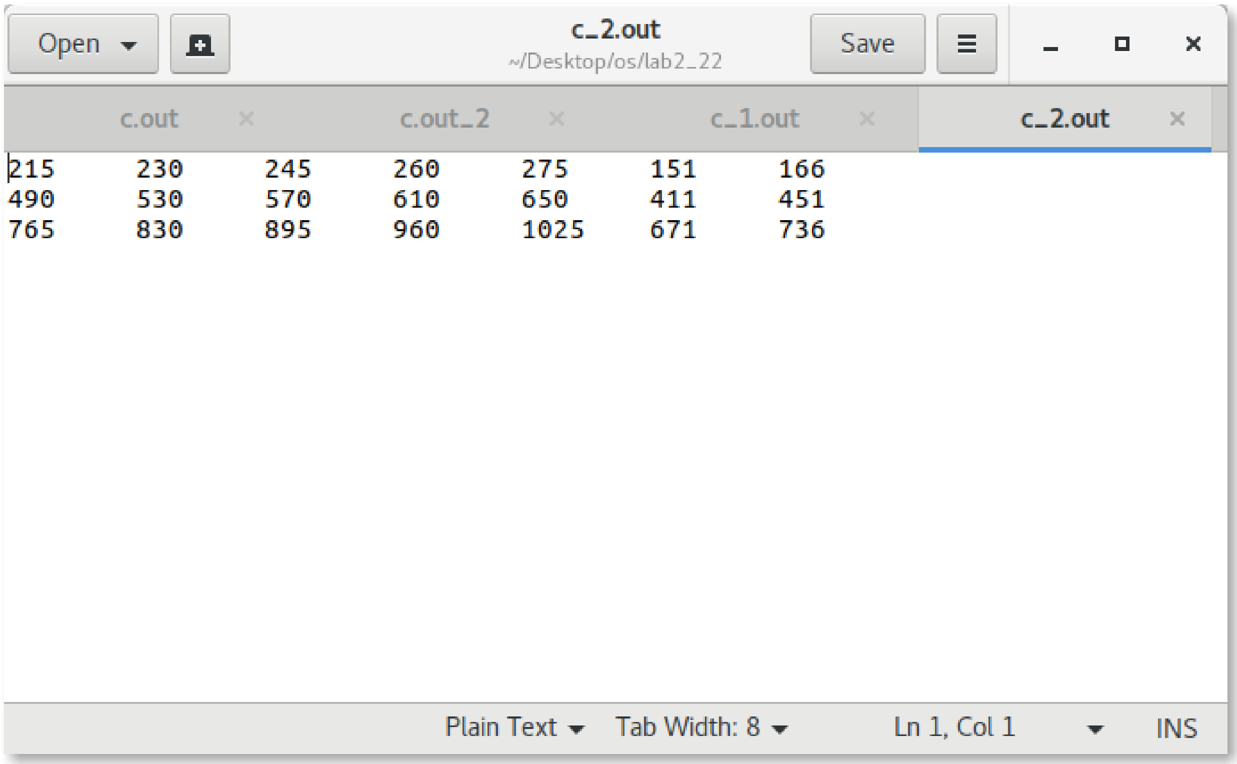


```
programajor@programajor-Inspiron-5537:~/Desktop/os/lab2_22$ ./matMultp
Row Method
Number of threads 3
Seconds taken 0
Microseconds taken: 142
Element Method
Number of threads 21
Seconds taken 0
Microseconds taken: 2169
programajor@programajor-Inspiron-5537:~/Desktop/os/lab2_22$
```

Running application with a (3x5) matrix and a (5x7) matrix from default files



```
215     230     245     260     275     151     166
490     530     570     610     650     411     451
765     830     895     960     1025    671     736
```

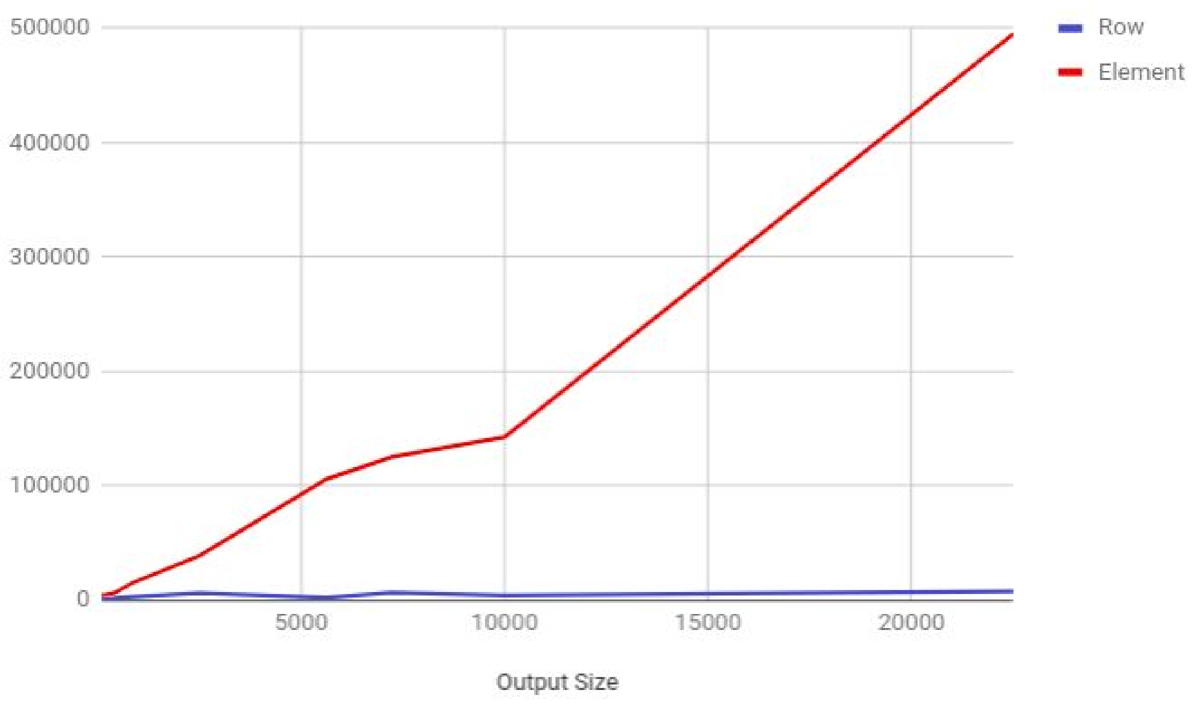


```
215     230     245     260     275     151     166
490     530     570     610     650     411     451
765     830     895     960     1025    671     736
```

Comparison between the two methods

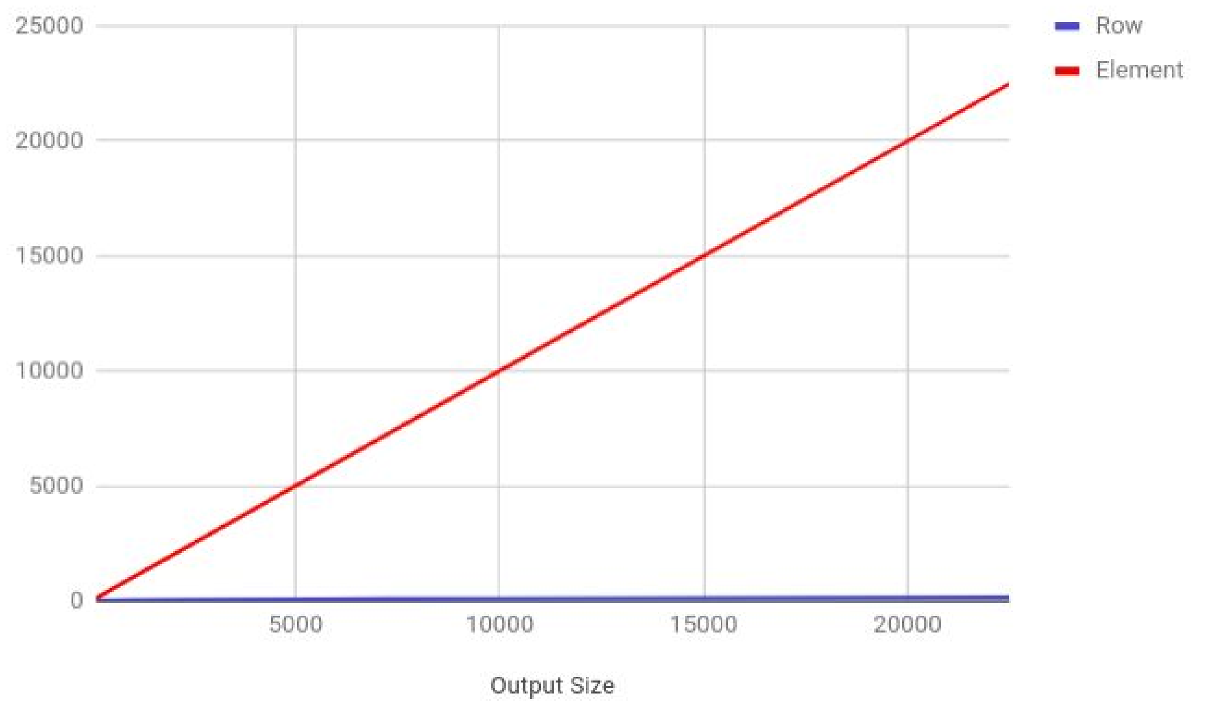
I- TIme in microseconds

Output Size	100	225	400	900	2500	5625	7225	10000	22500
Row	1219	970	1242	2709	5632	1929	6144	3762	7220
Element	4214	4733	5749	15612	38561	105539	124917	142317	494938



II- Number of threads

Output Size	100	225	400	900	2500	5625	7225	10000	22500
Row	10	15	20	30	50	75	85	100	150
Element	100	225	400	900	2500	5625	7225	10000	22500



How To Compile & Run it

1. Open the terminal
2. CD to the project directory
3. Write “make clean”
4. Write “make”
5. Write “./matMultp” or Write “./matMultp input1 input2 output”