Alexandria University - Faculty of Engineering
Computer Systems Engineering Department

Lap Assignment 2 Report
Threads

Name: Abanoub Milad Nassief

Seat Num. : 6

Code Organization:

Functional programming paradiam is adapted. Arguments sent to the program call are checked and validated to determine the input files, output files and the chosen method to perform the matrix multiplication.
Program created the threads according to the chosen method and waits for them to finish then writes the result matrix in output file.
Time is measured in seconds, milliseconds and microseconds seperately.

Functions:

void write_matout()

write matrix c to output file

void * mul_row_method(void* threadarg)

a thread function that computes each row in the output C matrix

void * mul_element_method(void* threadarg)

a thread function that computes each element in the output C matrix

void operate_row_method()

creates the threads where each thread computes each row in the output C matrix

void operate_element_method()

creates the threads where each thread computes an element in the output C matrix

int initialize_options(int argc, const char * argv[])

initialize the file names associated with matrices A,B and C
by user's preferences or by defaults : Matrix A : a.txt, Matrix B : b.txt and Matrix C : c.out
argc : number of input arguments sent to program call
argv[] : input arguments sent to program call
return 0 : a thread computes each row in the output C matrix
return 1 : a thread computes each element in the output C matrix
else terminate invalid input

void extract_dim(char * line,int* dim1,int* dim2)

extracts the matrix dimenions from a string line

line : input string line

dim1 : output first dimention

dim2 : output second dimention

void initialize_matrices()

initialize the matrices A, B and C using dimenions x,y and z

allocates memeory for each matrix

void populate_matrices()

populate the matrices A and B using pre-defined input files

void operate()

calls the procdure functions

Compiling and running matmult:

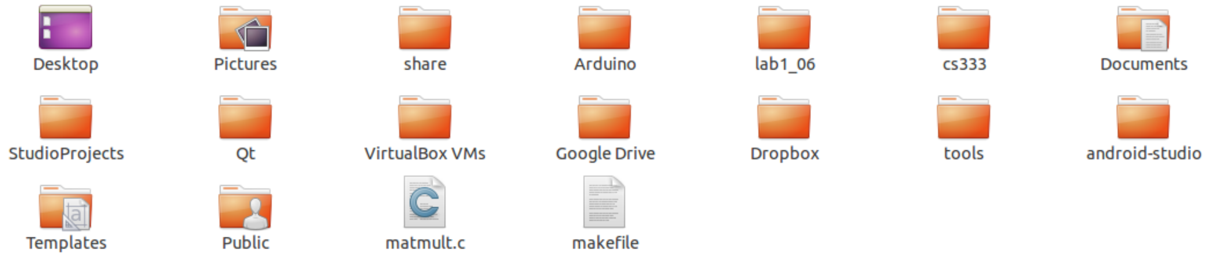
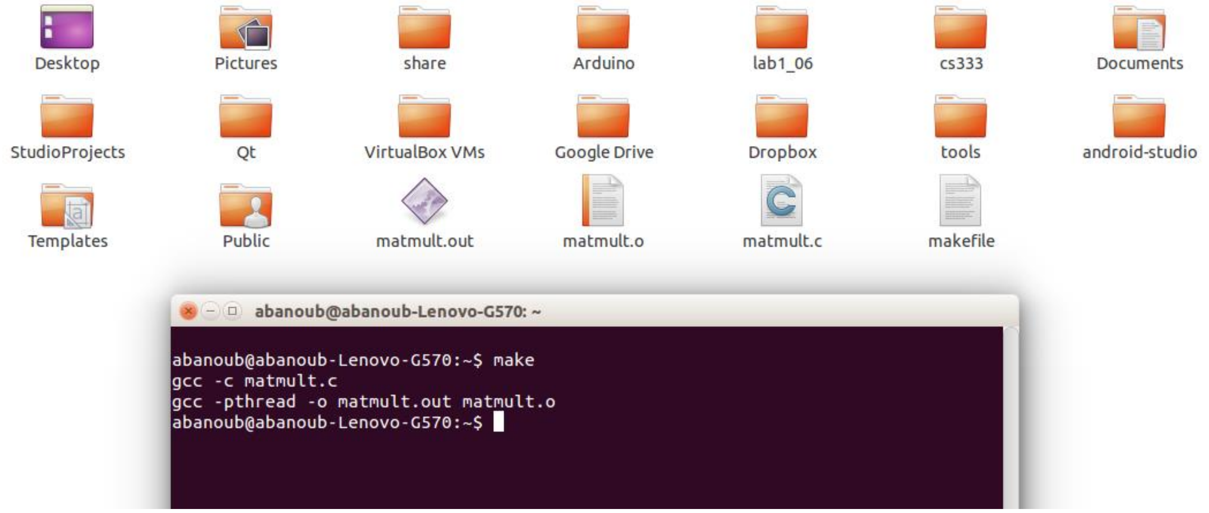
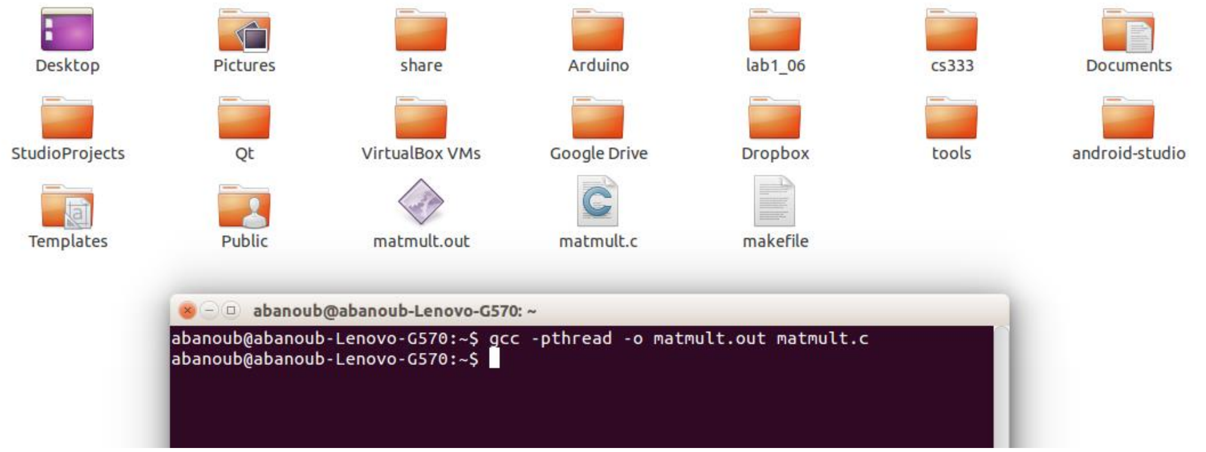


Figure 1 source code and makefile



```
abanoub@abanoub-Lenovo-G570:~$ make
gcc -c matmult.c
gcc -pthread -o matmult.out matmult.o
abanoub@abanoub-Lenovo-G570:~$
```

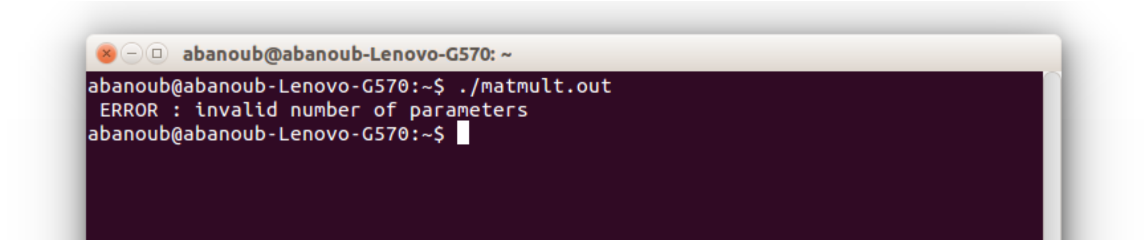
Figure 2 first method : running make command



```
abanoub@abanoub-Lenovo-G570:~$ gcc -pthread -o matmult.out matmult.c
abanoub@abanoub-Lenovo-G570:~$ 
```

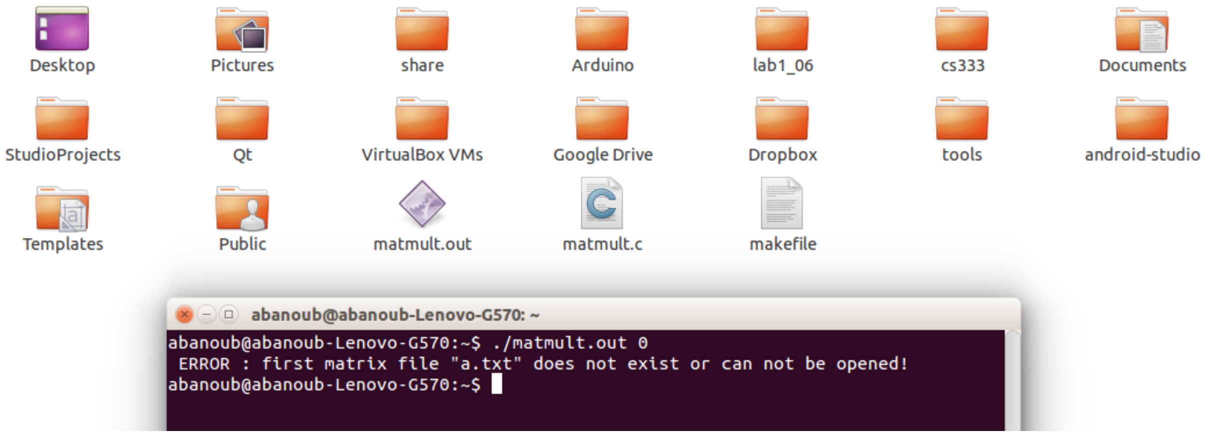
Figure 3 second method : compiling code directly

Sample Runs:



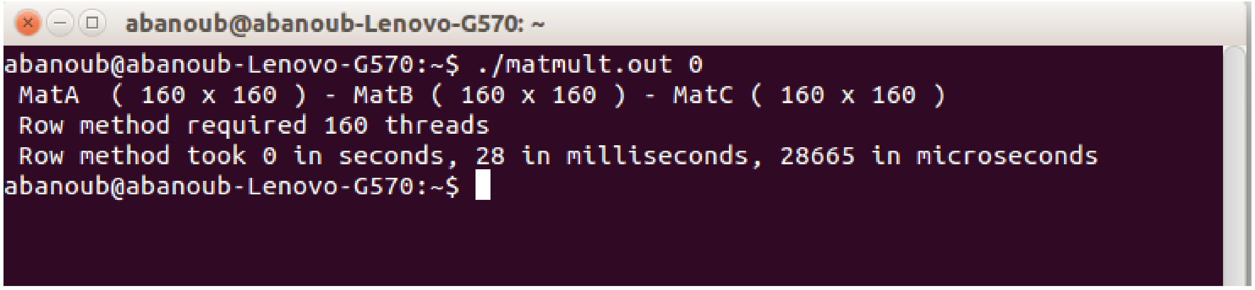
```
abanoub@abanoub-Lenovo-G570:~$ ./matmult.out
 ERROR : invalid number of parameters
abanoub@abanoub-Lenovo-G570:~$ 
```

Figure 4 calling executable with no arguments



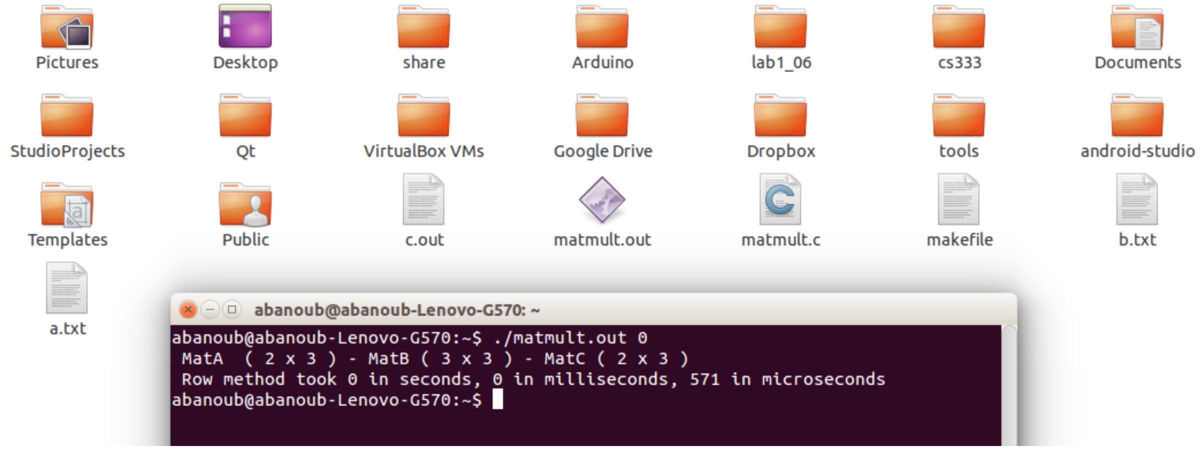
```
abanoub@abanoub-Lenovo-G570:~$ ./matmult.out 0
 ERROR : first matrix file "a.txt" does not exist or can not be opened!
abanoub@abanoub-Lenovo-G570:~$ 
```

Figure 5 calling executable with method type only (default options) while a.txt and b.txt don’t exist



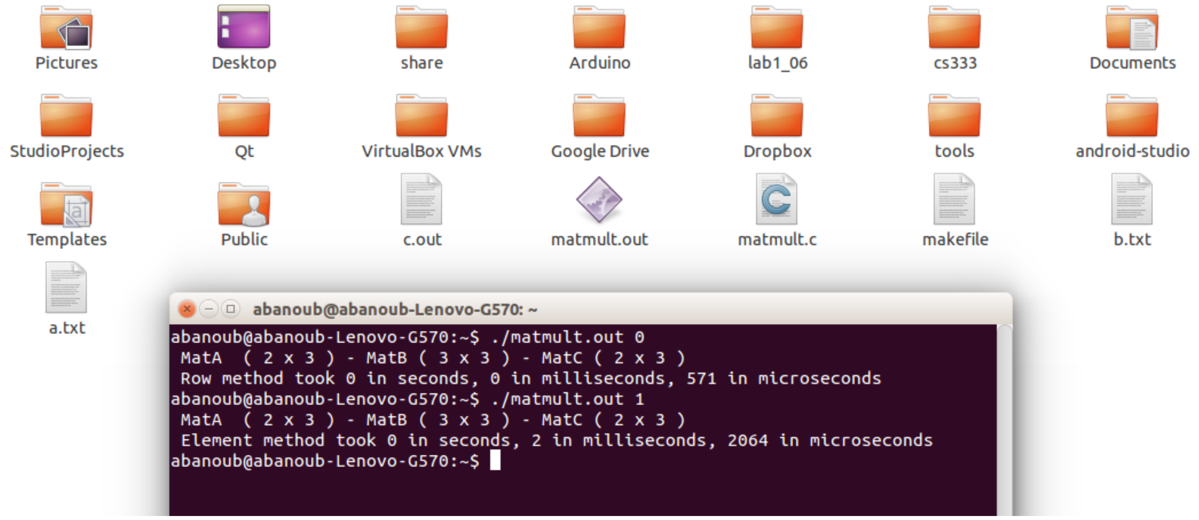
```
abanoub@abanoub-Lenovo-G570:~$ ./matmult.out 0
 MatA  ( 160 x 160 ) - MatB ( 160 x 160 ) - MatC ( 160 x 160 )
 Row method required 160 threads
 Row method took 0 in seconds, 28 in milliseconds, 28665 in microseconds
abanoub@abanoub-Lenovo-G570:~$ 
```

Figure 6 running row method - number of created threads is printed



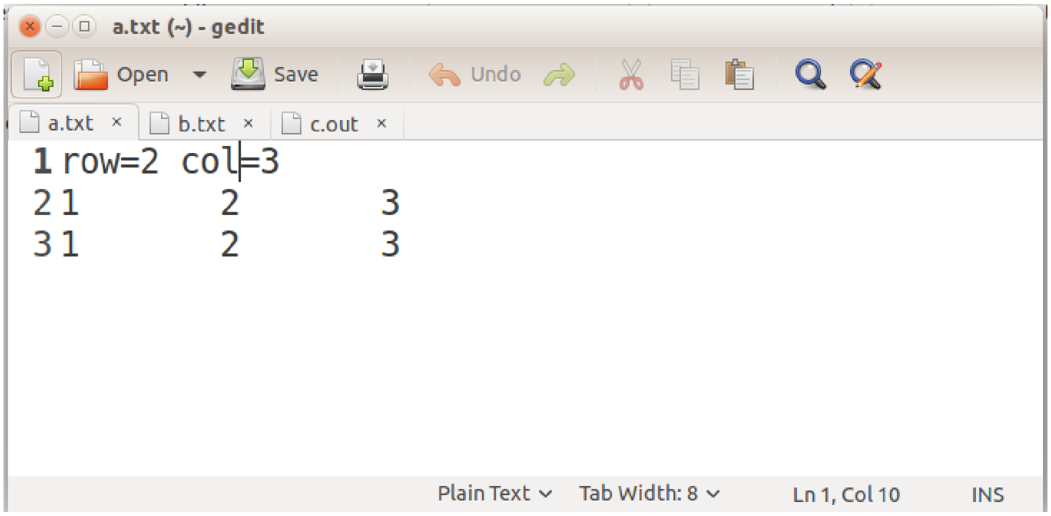
```
abanoub@abanoub-Lenovo-G570:~$ ./matmult.out 0
 MatA  ( 2 x 3 ) - MatB ( 3 x 3 ) - MatC ( 2 x 3 )
 Row method took 0 in seconds, 0 in milliseconds, 571 in microseconds
abanoub@abanoub-Lenovo-G570:~$ 
```

Figure 7 running row method in default mode, input files exist



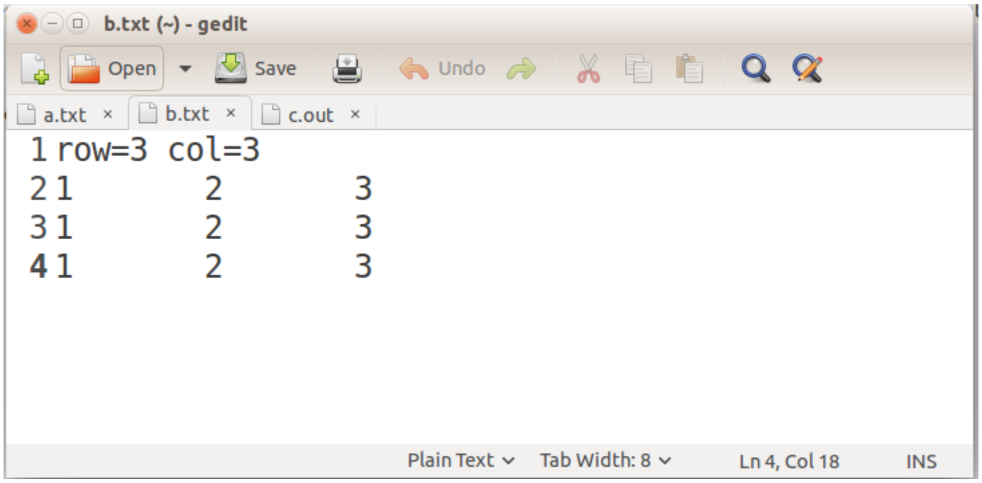
```
abanoub@abanoub-Lenovo-G570:~$ ./matmult.out 0
 MatA  ( 2 x 3 ) - MatB ( 3 x 3 ) - MatC ( 2 x 3 )
 Row method took 0 in seconds, 0 in milliseconds, 571 in microseconds
abanoub@abanoub-Lenovo-G570:~$ ./matmult.out 1
 MatA  ( 2 x 3 ) - MatB ( 3 x 3 ) - MatC ( 2 x 3 )
 Element method took 0 in seconds, 2 in milliseconds, 2064 in microseconds
abanoub@abanoub-Lenovo-G570:~$ 
```

Figure 8 running element method in default mode, input files exist



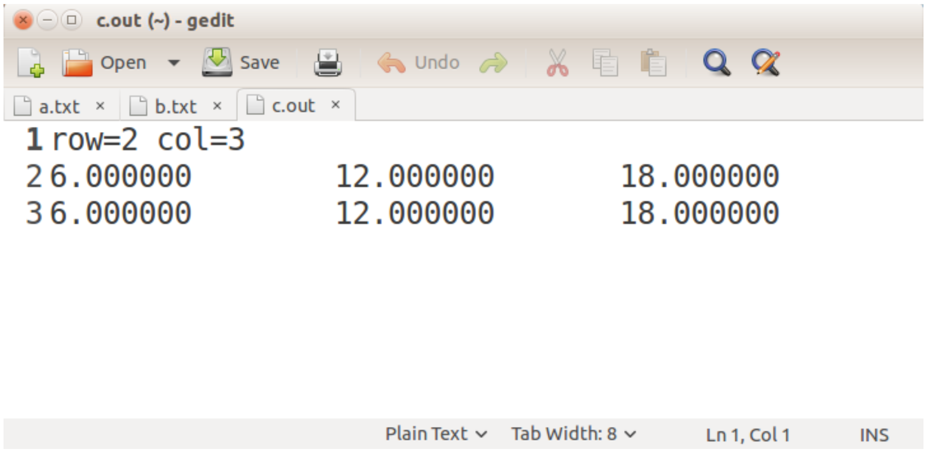
```
row=2 col=3
1       2       3
1       2       3
```

Figure 9 input file a.txt



```
row=3 col=3
1       2       3
1       2       3
1       2       3
```

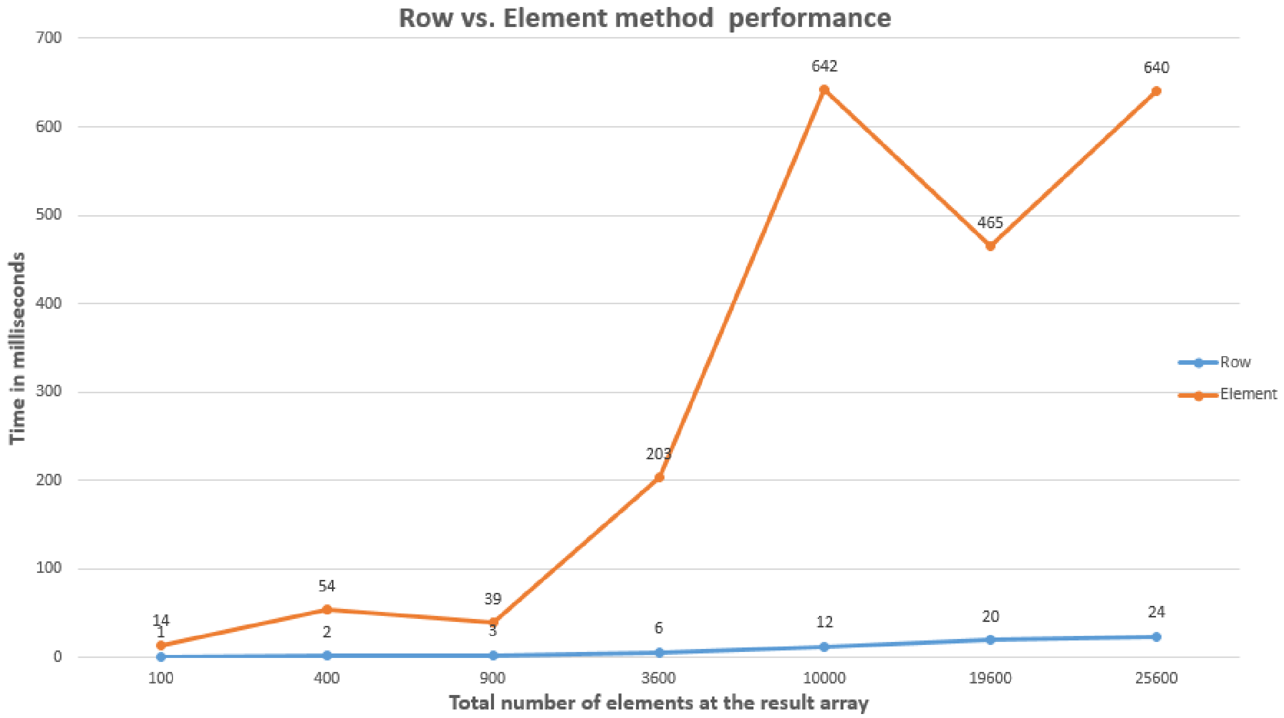
Figure 10 input file b.txt

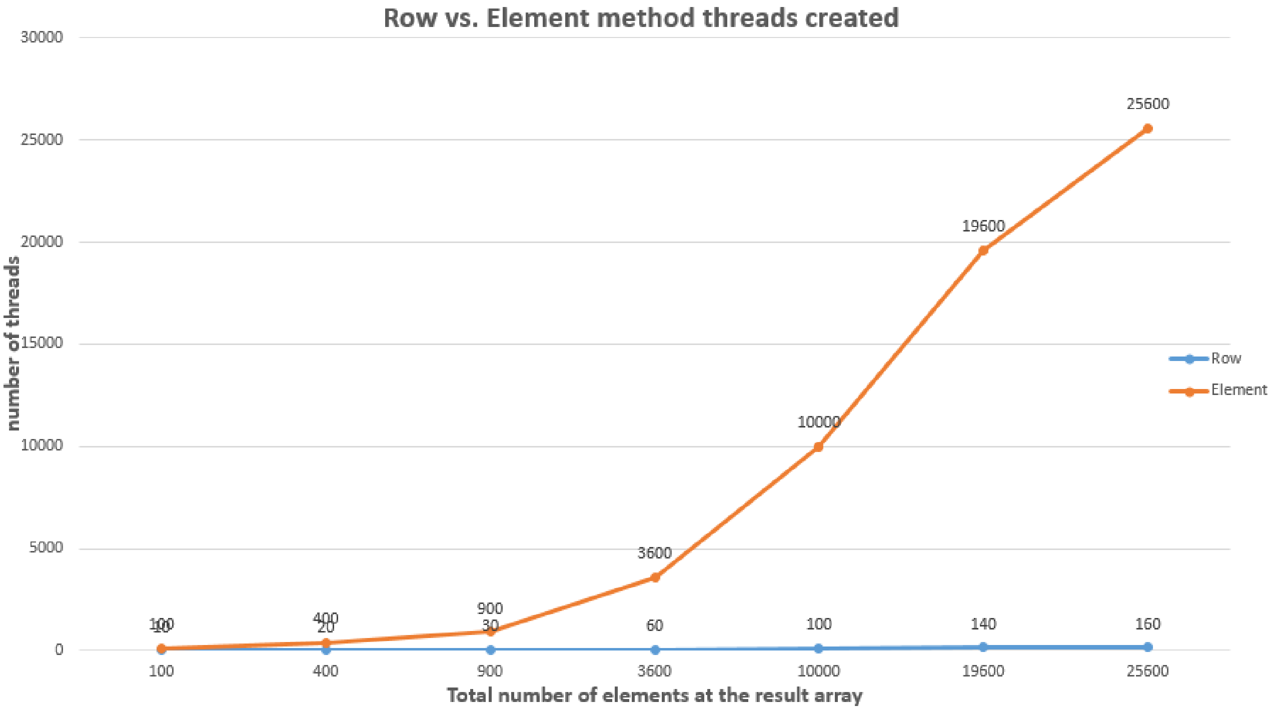


```
row=2 col=3
6.000000        12.000000       18.000000
6.000000        12.000000       18.000000
```

Figure 11 ouput file c.out

Row and element method Comparison:


Row vs. Element method performance
Time in milliseconds
Total number of elements at the result array
Row
Element


Row vs. Element method threads created
number of threads
Total number of elements at the result array
Row
Element