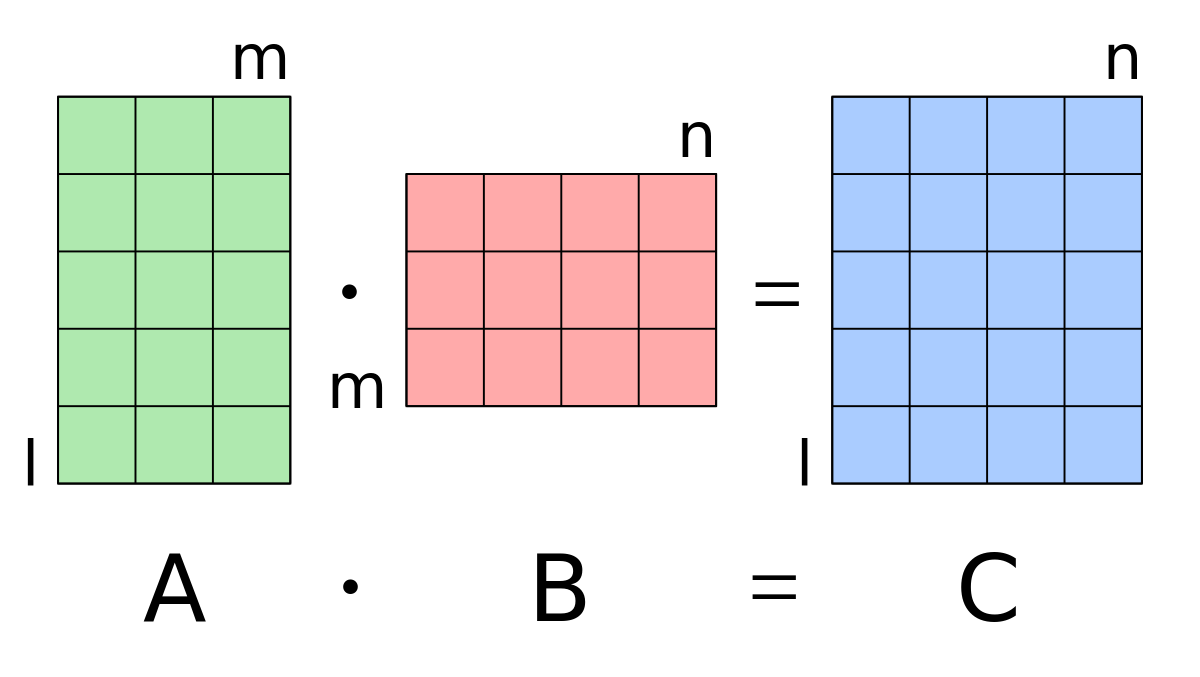
**Operating System**

**Lab 2**

# **Matrix Multiplication (Multi-Threading)**



Name : Mohamed Mohamed Hawas

ID : 18011595

## 

## **Objectives**

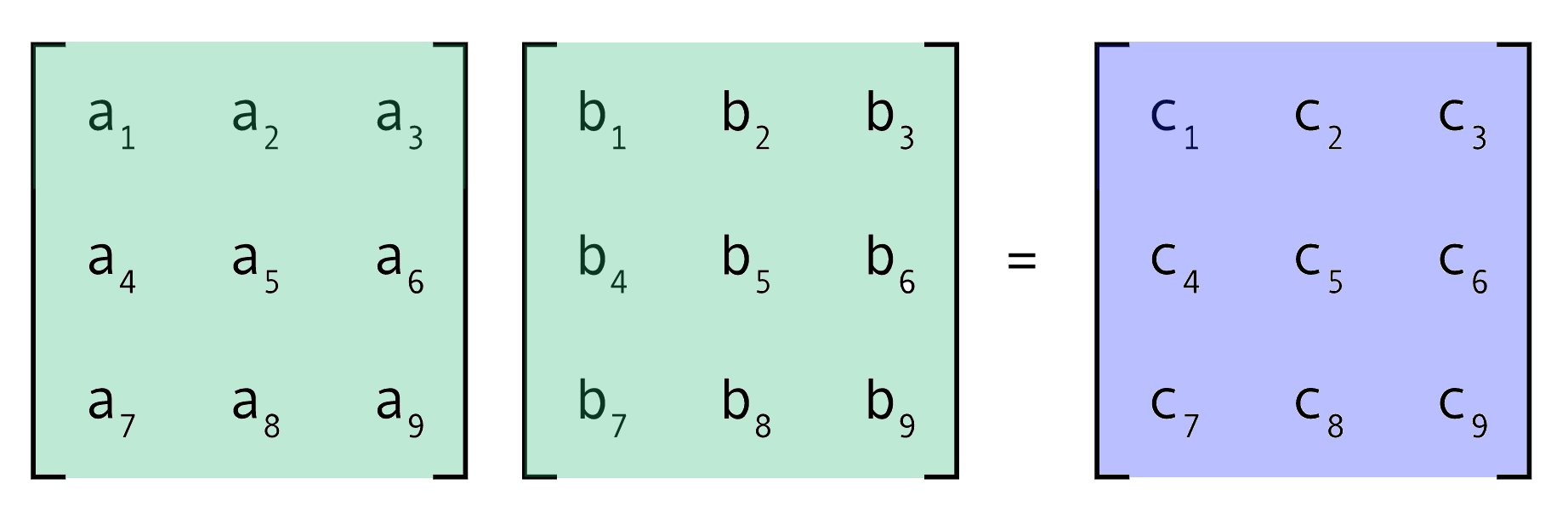
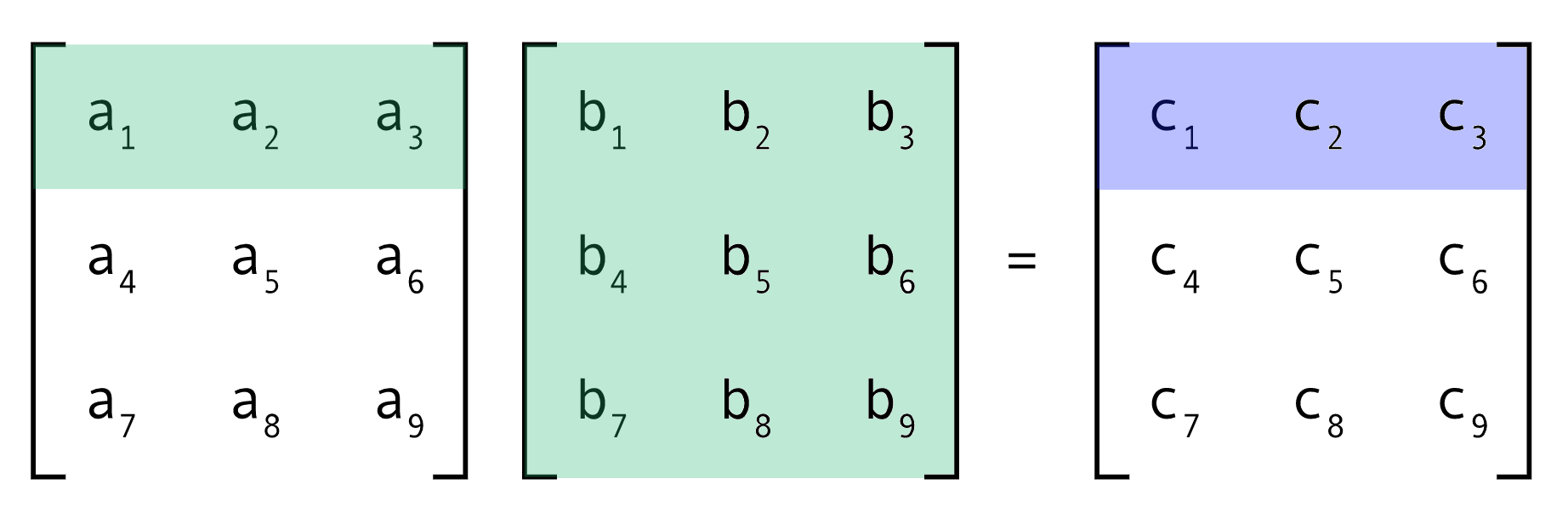
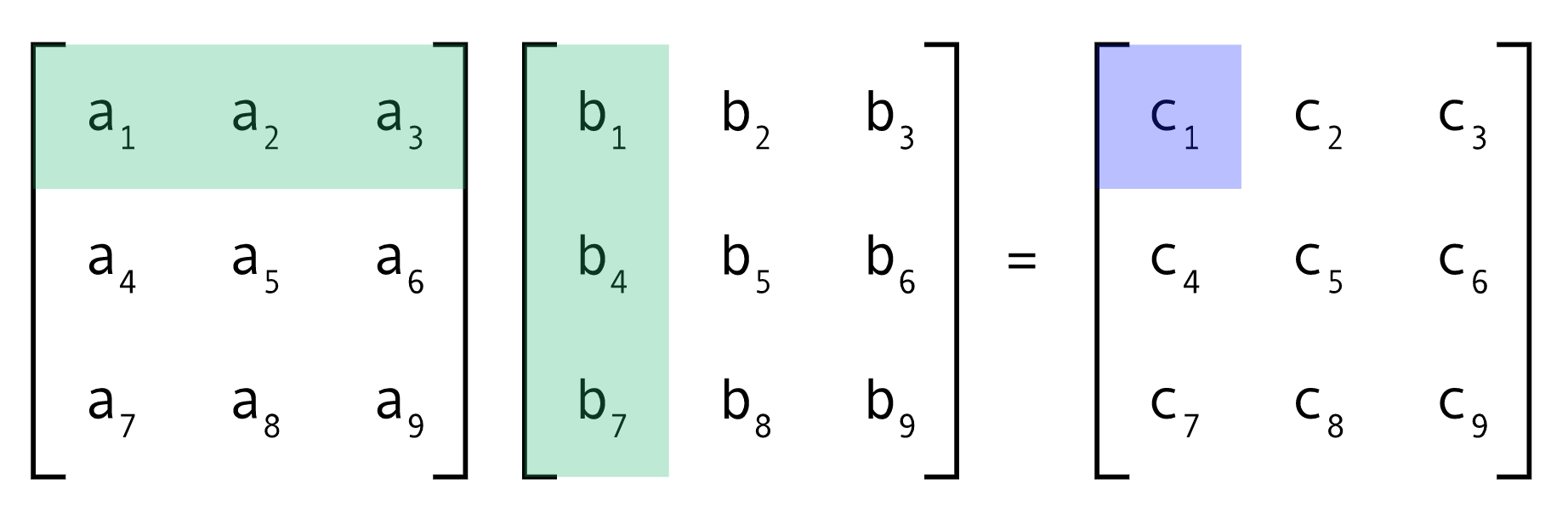
* To get familiar with thread programming using the Pthread library.
* To better understand processes and threads.

## **2. Overview**

You are required to implement a multithreaded matrix multiplication program.

The input to the program is two matrixes A(x*y) and B(y*z) that are read from corresponding text files. The output is a matrix C(x\*z) that is written to an output text file.

A parallelized version of matrix multiplication can be done using one of these three methods:

1. A thread computes the output C matrix i.e. without multi-threading. (A thread per matrix). 
2. A thread computes each row in the output C matrix. (A thread per row). 
3. A thread computes each element in the output C matrix. (A thread per element). 

## **3. Code Organization**

The program reads two matrices from files, multiplies them using three different methods, and writes the resulting matrices to output files. The program uses “pthreads” to parallelize the multiplication process. Here is an overview of how the code is organized:

1. The program defines a Matrix struct that holds the dimensions and elements of a matrix.
2. It defines three instances of Matrix: matrix\_a, matrix\_b, and matrix\_c.
3. The program defines three functions to read the contents of the matrices from files and populate the Matrix structs. These functions are read\_file, construct, and write\_file.
4. The program defines three multiplication functions:

* mult\_per\_matrix: multiplies matrix\_a and matrix\_b using the matrix multiplication algorithm and stores the result in matrix\_c\_per\_matrix.
* mult\_per\_row: multiplies matrix\_a and matrix\_b using the matrix multiplication algorithm and stores the result in matrix\_c\_per\_row. This function is designed to be run by multiple threads, with each thread computing one row of the result matrix.
* mult\_per\_element: multiplies matrix\_a and matrix\_b using the matrix multiplication algorithm and stores the result in matrix\_c\_per\_element. This function is designed to be run by multiple threads, with each thread computing one element of the result matrix.

1. The main function reads the matrices from files using read\_file, constructs the matrix\_c matrices using construct, and creates the threads to perform the multiplications using pthread\_create.
2. After the threads are created, the main function waits for all threads to complete using pthread\_join.
3. Finally, the main function writes the resulting matrices to output files using write\_file.

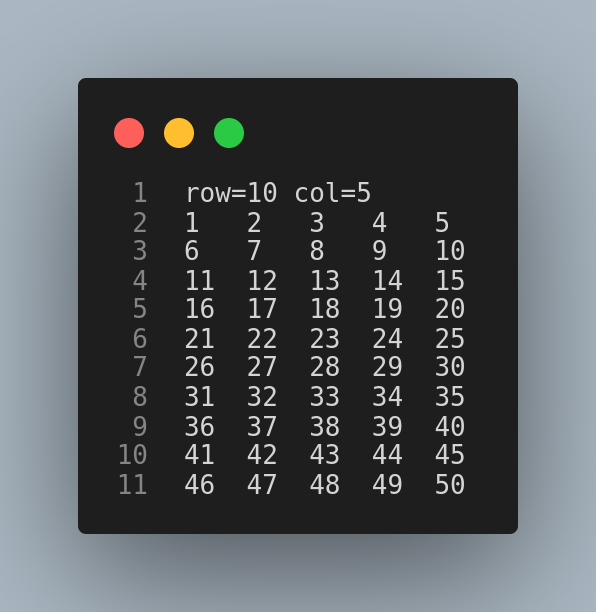
## **4. Code main functions**

1. **construct(Matrix\* matrix, int rows, int cols)**: A function that constructs a matrix with the given number of rows and columns.
2. **void\* read\_file(void\* arg, Matrix\* matrix)**: A thread function that reads the matrix data from a file and stores it in the given matrix.
3. **void write\_file(Matrix\* matrix, char\* filename)**: A function that writes the matrix data to a file.
4. **void\* mult\_per\_matrix():** A thread function that multiplies two matrices (matrix\_a and matrix\_b) and stores the result in matrix\_c\_per\_matrix.
5. **void\* mult\_per\_row(void\* arg)**: A thread function that multiplies two matrices (matrix\_a and matrix\_b) row by row and stores the result in matrix\_c\_per\_row.
6. **void\* mult\_per\_element(void\* arg)**: A thread function that multiplies two matrices (matrix\_a and matrix\_b) element by element and stores the result in matrix\_c\_per\_element.

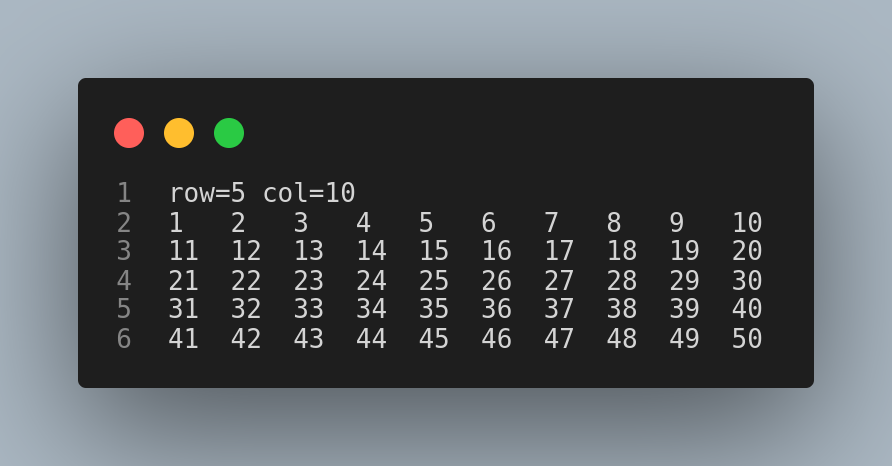
## **3. Sample Runs and comparison**

## 

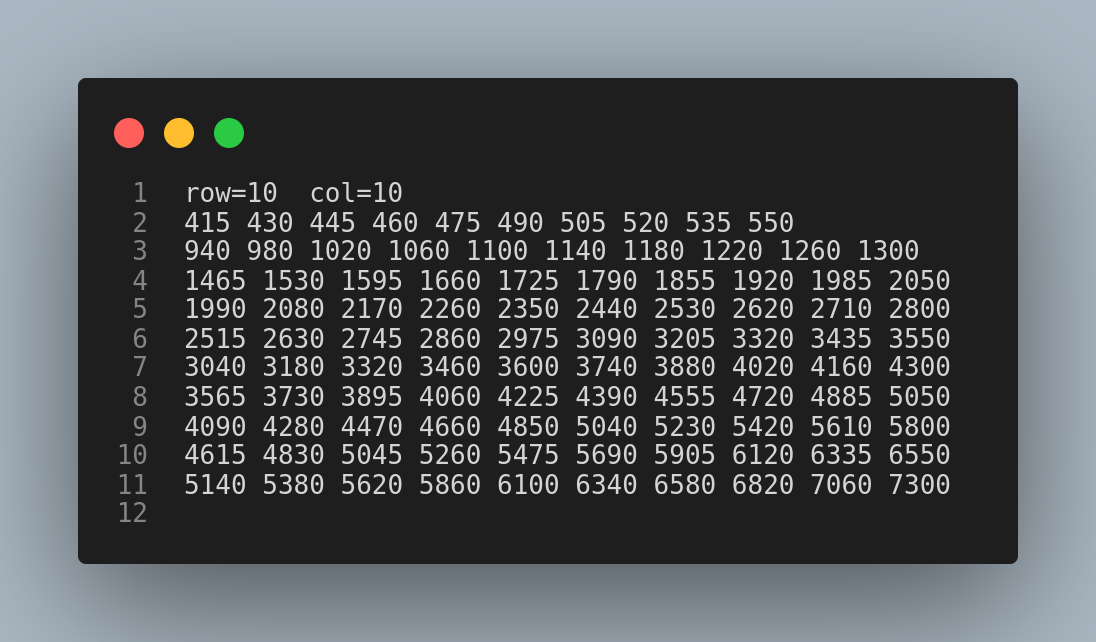
* **a\_txt**



* **b\_txt**

****

* **c\_txt (For all)**

****