Week 8 - Assignment (2 marks)

Note: All programs must use the appropriate C++ features.

Objective:

- Implement a robust matrix class using object-oriented programming principles in C++.
- Apply multithreading **std::thread** to enhance computational efficiency for large matrix multiplication.
- Compare the performance of single-threaded and multithreaded computations.

Tasks: Design Matrix Class for Matrix Multiplication

1. Design and implement a Matrix class that supports dynamic memory management to handle large matrices.
2. The class should include **default constructors** and **destructor**.
3. The default constructor is used to initialize a matrix with 0.
4. A member function **fill_rand_value** to initialize a matrix with random values**.**
5. Include methods for matrix multiplication in mathematics, ensuring the class can handle non-square matrices as well.

   If $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{B}$ is an $n \times p$ matrix,

   $$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix}$$

   the *matrix product* $\mathbf{C} = \mathbf{AB}$ (denoted without multiplication signs or dots) is defined to be the $m \times p$ matrix[5][6][7][8]

   $$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{pmatrix}$$

   such that

   $$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj},$$

   for $i = 1, \ldots, m$ and $j = 1, \ldots, p$.

6. **Single-threaded Multiplication Implementation**
   a. Implement the standard matrix multiplication algorithm in a single-threaded manner within the Matrix class.
   b. Ensure the method handles exceptions (e.g., mismatched size) correctly.
7. **Multithreaded Multiplication Implementation**

a. Extend the matrix multiplication method to use multithreading, where each thread computes a portion of the result matrix.
b. Consider dividing the workload by rows or blocks of rows to distribute the multiplication tasks among threads.
c. Ensure the method handles exceptions (e.g., mismatched size) correctly.

The input and output should be like below:

Input:

Based on the given **week8.cpp** to develop your program. Must not change the supplied code.

Output

Test at least three cases, as shown below. Please note that the running time may vary depending on the specifications of different computers.

```
Enter the number of rows of Matrix A: 10
Enter the number of cols of Matrix A: 20
Enter the number of rows of Matrix B: 10
Enter the number of cols of Matrix B: 20
Enter the number of threads: 2
Matrix dimensions must match!
Quit (Y/N): n

Enter the number of rows of Matrix A: 100
Enter the number of cols of Matrix A: 50
Enter the number of rows of Matrix B: 50
Enter the number of cols of Matrix B: 100
Enter the number of threads: 2
Single-threaded multiplication took 3.355 ms.
Multithreaded multiplication took 18.6884 ms.
Validating results...
Results are identical!
Quit (Y/N): n

Enter the number of rows of Matrix A: 500
Enter the number of cols of Matrix A: 500
Enter the number of rows of Matrix B: 500
Enter the number of cols of Matrix B: 500
Enter the number of threads: 2
Single-threaded multiplication took 973.992 ms.
Multithreaded multiplication took 482.789 ms.
Validating results...
Results are identical!
Quit (Y/N): y
```

Submit:

1, **all C++ source code**: *.cpp and *.hpp if your code is organized into separate files.

Organizing the source code into separate files is not mandatory.

You can consolidate all code into a single cpp file.

2, **week8.txt**: a txt file contains all the source code.

3, **output.jpg**, or output.png, or output.bmp: a screenshot of the output by your program

Please refer to the submission page for the Marking Rubric.