Assignment 1
Matrix Multiplication by Partitioning

Linear Algebra and Random Processes (CS6015)
Input/Output Specifications and Coding Description

Your program (matrix_mul.c/matrix_mul.cpp) written in C/C++ programming language, should read the files given under input section and generate the files specified under output section. All the files should be present at the same directory level.

1 Input for your program

1. matA.txt - Matrix $A \in \mathbb{R}^{m \times n}$
2. matB.txt - Matrix $B \in \mathbb{R}^{n \times p}$

2 Output generated by your program

1. matC_N.txt - Product computed using naive method
2. matC_P.txt - Product computed by partitioning (Single level)
3. matC_RP.txt - Product computed by recursive partitioning
4. output.txt - File containing the outputs described in the problem statement

3 Matrix File Format

The first line contains two space-separated integers m, n. Each of the m subsequent lines contains n space-separated floating point numbers, denoting the matrix $A \in \mathbb{R}^{m \times n}$.

This format should be used for files "matA.txt", "matB.txt", "matC_N.txt", "matC_P.txt" and "matC_RP.txt"

3.1 Sample

2 3
1.0 2.0 3.0
4.0 5.0 6.0

4 Output File Format

This file contains four lines.
The first line contains the time taken to compute product using naive method.
The second line contains the time taken to compute product by partitioning (Single level).
The third line contains the time taken to compute product by recursive partitioning.
The fourth line contains the Frobenius norm of the difference between the matrix product computed using the naive method and the product computed using recursive partitioning.

All values representing time are measured in seconds (use double precision). The Frobenius norm is represented as a double precision floating point number. This format should be used for the file "output.txt".

The time elapsed can be computed using the following code snippet.

```
#include <time.h>

clock_t start, end;
double cpu_time_used;

start = clock();
... /* Code to be timed. */
end = clock();
cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
```

4.1 Sample Output

1240.956
923.948
623.948
0.123456