Parallel Computing homework

1.Build project:

There are 2 ways to build the c file : makefile or Cmake.

+ build c file with makefile (this method uses gcc compiler):

- Step 1 : Open terminal on the file directory contains the source code matrixAverage.c
- Step 2: In terminal run comment: make
- The makefile will execute and run 2 comments:

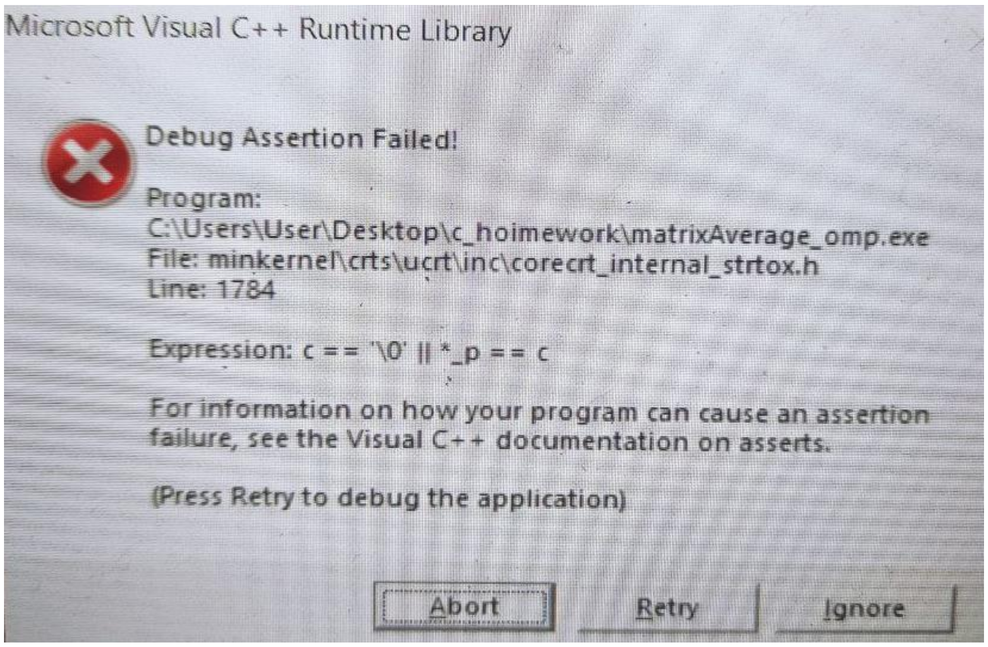
```
gcc matrixAverage.c -o matrixAverage
gcc -o matrixAverage_omp -fopenmp matrixAverage.c
```

matrixAverage.exe is sequential version and **matrixAverage_omp. exe** is parallel version.

+ To build project with CMake:

- Step 1: Open terminal on the file directory contains the source code matrixAverage.c
- Step 2: In terminal run: cmake -S . -B build
- Step3: In terminal run: cd build
- Step 4: In terminal run : cmake – build .

CMake will generate **matrixAverage.exe** (parallel version) in Debug directory. If run from debug directory, it will cause error file not found because I use relative path, please move it to the root directory of the project to run it. I try to build it using virtual studio, it builds but error when running.

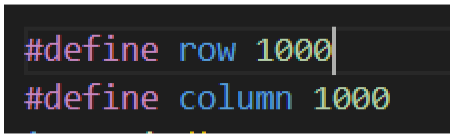


```
Microsoft Visual C++ Runtime Library

Debug Assertion Failed!

Program:
C:\Users\User\Desktop\c_hoimework\matrixAverage_omp.exe
File: minkernel\crts\ucrt\inc\corecrt_internal_strtox.h
Line: 1784

Expression: c == '\0' || *_p == c

For information on how your program can cause an assertion
failure, see the Visual C++ documentation on asserts.

(Press Retry to debug the application)
```

2. Generate Dataset:

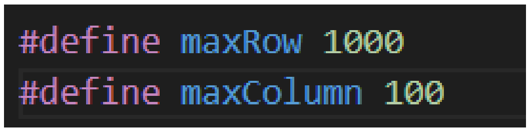
In this homework, I create a small program to generate matrix size 1000 x 1000. the program store in **createData.c**. it will generate float number between 0 to 10000000 and store in ".\input\test file\data.txt". please create the file data.txt if not exist.

To generate matrix with difference size, please change the value variable row and column in **createData.c**.



```
#define row 1000
#define column 1000
```

Please noted that, the number of row and column in **matrixAverage.c** should be the same as the dataset. If the dataset is bigger, we will not be able to read all the data, is smaller we will read all the data, the column that no data will store value 0.0 .



```
#define maxRow 1000
#define maxColumn 100
```

3. Program flow:

Sequential Program flow:

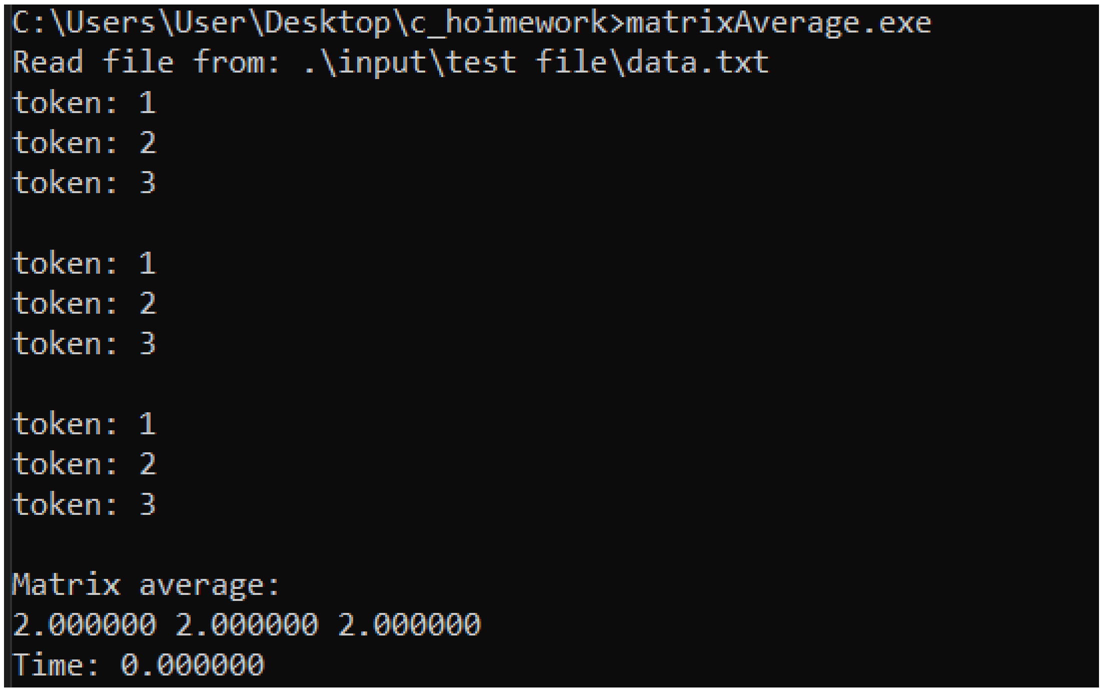
- Read Matrix one row at a time with loop
- In each row read one column at a time using loop
- Print out the column value (if not printing the column value, the program execute too fast and sometimes sequential code is faster than parallel code)
- Sum all the column and find average than store in the output vector
- Loop though all the element of the output vector and print the value

Parallel program flow:

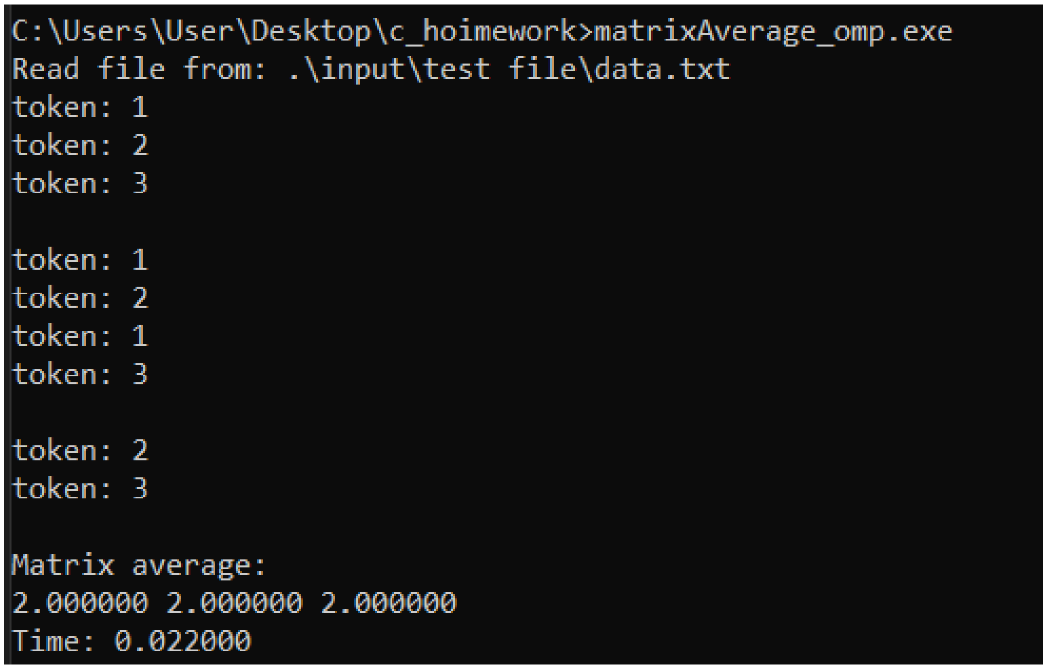
- Same with the sequential version
- The parallel part is at the first loop when reading the code one row at the time, we can read multiple rows and find the average then store it in the average vector. Element in the average vector will divided into section by the parallel code so no problem.
- the second loop is used for reading each column, find sum and average. I did not parallel this loop because I think sum is a share variable.
-

4. Testing:

Parallel version and sequential version give the same output (Matrix average) is small testing.



```
C:\Users\User\Desktop\c_hoimework>matrixAverage.exe
Read file from: .\input\test file\data.txt
token: 1
token: 2
token: 3

token: 1
token: 2
token: 3

token: 1
token: 2
token: 3

Matrix average:
2.000000 2.000000 2.000000
Time: 0.000000
```



```
C:\Users\User\Desktop\c_hoimework>matrixAverage_omp.exe
Read file from: .\input\test file\data.txt
token: 1
token: 2
token: 3

token: 1
token: 2
token: 1
token: 3

token: 2
token: 3

Matrix average:
2.000000 2.000000 2.000000
Time: 0.022000
```

5.Result:

The table below show the testing result in matrix 1000 x 1000. I think my program is not correct because the parallel version is slower than the sequential version.

No	Sequential	Parallel		
		Core 2	Core 3	Core 4
1	27.591	29.625	29.771	31.16
2	26.832	29.695	29.674	31.062
3	27.761	29.479	29.716	31.043
4	26.941	29.362	30.157	31.17
5	27.037	29.756	29.897	31.205
Average	27.2324	29.5834	29.843	31.128
Max	27.761	29.756	30.157	31.205