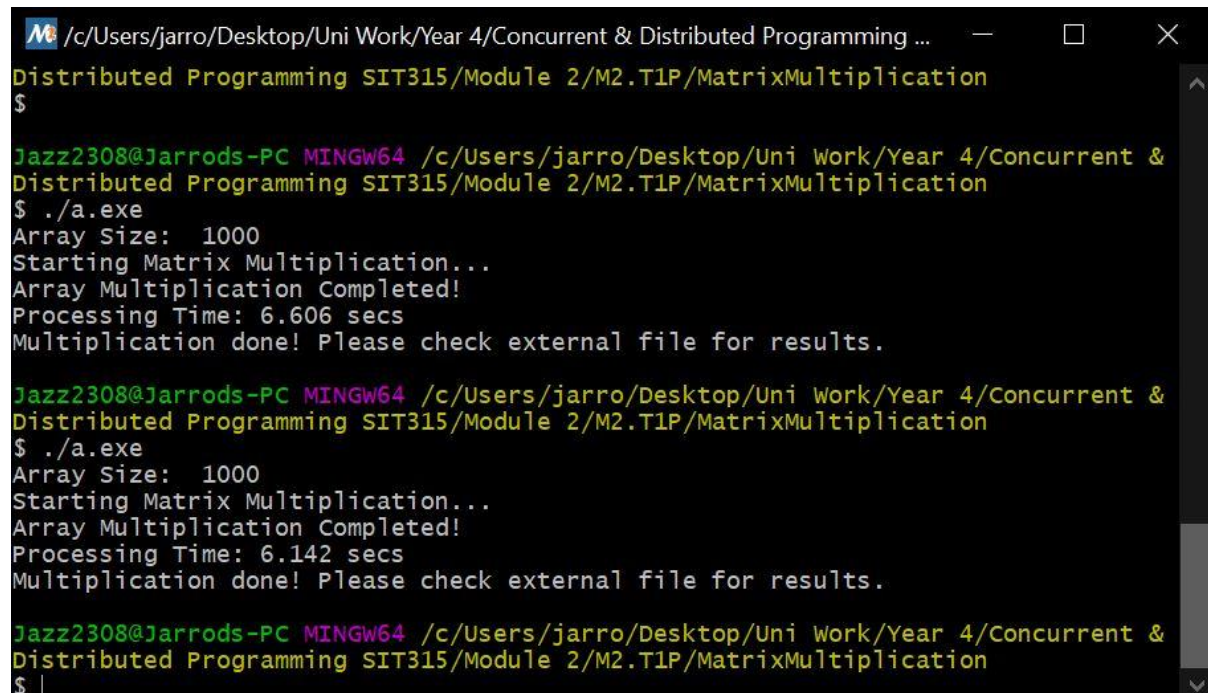


1. Program can be found in submission

2.



```

/c/Users/jarro/Desktop/Uni Work/Year 4/Concurrent & Distributed Programming ...
Distributed Programming SIT315/Module 2/M2.T1P/MatrixMultiplication
$
Jazz2308@Jarrods-PC MINGW64 /c/Users/jarro/Desktop/Uni Work/Year 4/Concurrent &
Distributed Programming SIT315/Module 2/M2.T1P/MatrixMultiplication
$ ./a.exe
Array Size: 1000
Starting Matrix Multiplication...
Array Multiplication Completed!
Processing Time: 6.606 secs
Multiplication done! Please check external file for results.

Jazz2308@Jarrods-PC MINGW64 /c/Users/jarro/Desktop/Uni Work/Year 4/Concurrent &
Distributed Programming SIT315/Module 2/M2.T1P/MatrixMultiplication
$ ./a.exe
Array Size: 1000
Starting Matrix Multiplication...
Array Multiplication Completed!
Processing Time: 6.142 secs
Multiplication done! Please check external file for results.

Jazz2308@Jarrods-PC MINGW64 /c/Users/jarro/Desktop/Uni Work/Year 4/Concurrent &
Distributed Programming SIT315/Module 2/M2.T1P/MatrixMultiplication
$ |
```

3. How should this program be modified?

After thorough research and analysis, the most common method to implement parallelisation in Matrix multiplication is by dividing the multiplication steps across multiple threads. All of the other functions are dependent on the input from the user or output from the matrix except the multiplication which is complex task, and each row and column can be multiplied separately which should not change the results.

So, by dividing the steps in multiplication of the matrices such that all the threads would have equal or close to equal number of elements to operate using a for loop. When all threads are run simultaneously, the speed of the program should improve significantly.

4. Threading program can be found on the link [here](#). Pthread library couldn't be used due to system issues so Thread was used instead.

5. Evaluation of the program: The program was significantly faster than sequential program and completed the process successfully. However, the speed didn't really had any improvements as the number of threads were increased after 2 to 4.

6. Program for OpenMP Version in submission

7. Using OpenMP did not see any improvements. Threading was the faster option.