COM6521: Assignment 1
Gang Chen

FINAL Mark (100%)	46	100

Lateness penalties are applied to this mark by the teaching office.

Part 1: Code (50%)

Is it functionally correct?; Has iterative improvement produced a sufficiently optimised final program?; Does the code make good use of OpenMP?; Is memory handled appropriately and efficienty?; Are there compiler warnings?; Is the structure clear and easy to parse?
A low mark suggests that you need to think more carefully about what is required to write and optimise C and OpenMP code; A mark in the middle of the range suggests that you have some understanding of how to use C and OpenMP. A high mark suggests you are well on your way to understanding how to use C and OpenMP to produce efficient optimised code.

24 50

Functionally Correct

4 functional tests failed.
See page 2 'What needs to be done to make it better?' for detail.

Iterative Improvement

Performance could be improved.

Memory Bandwidth

Image average calculation duplicates work of mosaic tile average's.

OpenMP Race Conditions

No substantial OpenMP used, code suggests lack of understanding of OpenMP for statement.

OpenMP Variable Scoping

No substantial OpenMP used.

Memory Accesses & Warnings

Compiler warnings should be actioned as they may point to bugs.
All memory allocated with malloc() should be free()'d before the program exits.

Robust Input Handling

7 robustness tests failed.
Your program should ensure all user input is valid, exiting with a graceful message if unexpected data is detected.

Well Written Code

Code requires greater commentary and division into methods (to reduce code duplication).

Part 2: Documentation (50%)

Is there a description of the implemntation? Is there a justifcation of the final implementation, including any steps taken to reach that conclusion? Description of the general testing process? Has performance throughout development been analysed – is there evidence of this?
A low mark suggests that you need to think more carefully about what is requiredto document development and optimisation. A mark in the middle of the range suggests that you have some understanding of the process and some consideration of attention to detail. A high mark suggests you are well on your way to understanding how to document the optimisation of code you develop.

22 50

Description of CPU Implementation

Detailed explanation provided with some justifications

Description of OpenMP Implementation

Should implement OMP parallel for statements rather than sections. Given the similarity in timings, I am wondering if `#pragma omp section` is not doing anything at all.

Iterative Improvement

More evidence of iterative improvements could be given. There should be reasoning for what aspects to parallelise by using theory and discussion

Analysis of Optimisation

Nice to see VS profiling tools being used

Overall

What is good about this work?

A functional implementation of the CPU mosaic algorithm. Some good descriptions in report.

What needs to be done to make it better?

Functional tests failed:

* No file was output when the optional output format arg was not provided
* When C was not a factor of image width the output image had some unusual artefacts.
* Larger images caused image colour average values to be wrong (presumed integer overflow).

Consider the cost of launching OpenMP blocks for small bits of code.

Refer to lectures and lab classes for how to implement an OpenMP parallel for statement

Robustness testing may provide bad program arguments or PPM files.

Try to address compiler warnings as they identify where the compiler had to make assumptions about your intentions.

All manually allocated memory must be free'd before the program exits.

Report could use more analysis and reasoning when discussing various topics