**GPU, Many-core and Cluster Computing Assignment**

1.

I profiled the code using the gprof performance analysis tool. With initially no optimisation (so compiled with the line *g++ -O0 –p –g karman.cpp –o karman*) the code ran very slowly, even for a time cut off variable of 0.1. In this case the majority of the time was spent in the *getCellIndex* and *assertion* functions, about 74% of the time in fact. This is because of the large number of times that the *getCellIndex* function is called, which in turn calls the *assertion* function.

After some compiler optimization (by doing *g++ -O3 –p –g karman.cpp –o karman*) the run speed was faster. Looking at the profiling information the bottleneck now is in the functions *computeP* and *setPressureBoundayConditions* which take up a total of about 85% and 15% of the running time respectively.

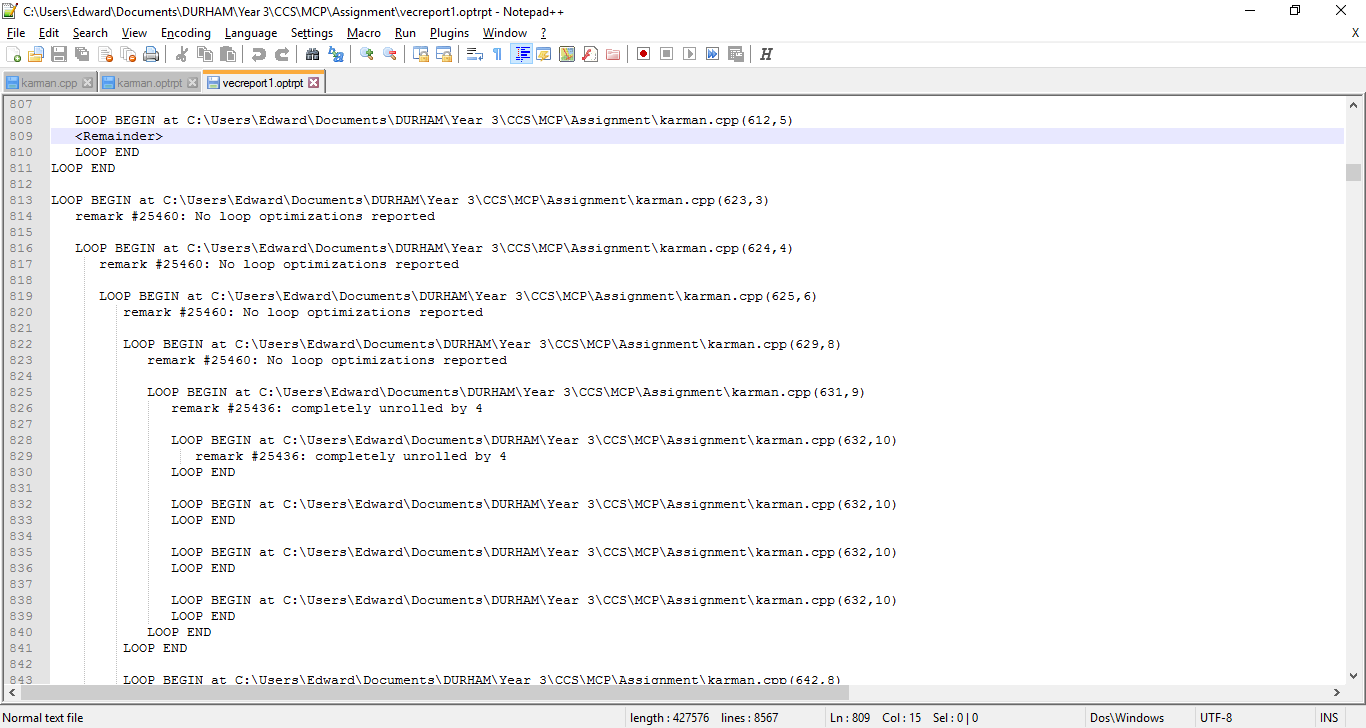
The functions that cause the issues when there was no optimization no longer cause any issues. This is because the compiler can tell that the *assertion* function doesn’t actually do anything since CheckVariableValues is not defined, and since *getCellIndex* just called *assertion* multiple times then returns a value, that function call can just be replaced by its return value.

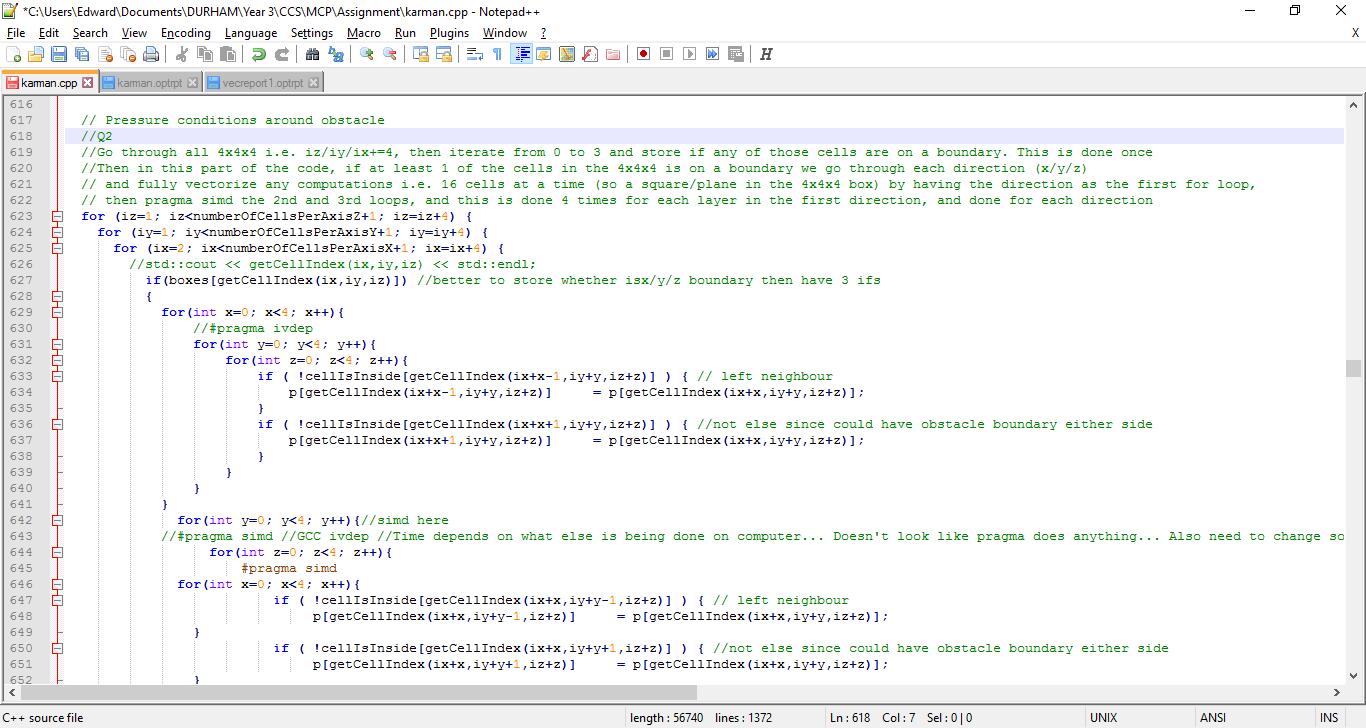
THIS IS NOT FINISHED

2.

The reason the output is different is due to the order in which the boundary values are assigned. E.g. for an edge bit since 2 cells will write, pressure is dependent on which neighbour writes last

Explain box thing

From the intel vectorization report:

The above image is the vectorization report of the below code. See how at line 629 there is no vectorization, but at line 631 there is. This is without the explicit pragma included, showing it was automatically done by the intel compiler –O3 option.

To generate vec report do “icl –O3 /Qopt-report:1 karman.cpp –o karman”