teamName: Cheesecake
student1_fullName: Andrei Patranoiu
student1_studentNumber: 998130696
student1_email: andrei.patranoiu@mail.utoronto.ca
student2_fullName: He Mutian
student2_studentNumber: 1004654475
student2_email: mutian.he@mail.utoronto.ca

Multi-Threading:

Since the test server has 8 hyperthreaded cores we wanted to make full use of them by implementing a 16 thread implementation of GoL. We break up the board into 16 equally sized pieces and assign each of these miniboards to one of the threads for processing. Since each iteration of the GoL algorithm requires the results of the previous iteration to proceed we couldn't just let the threads run unsynchronized as this would produce skewed results. To synchronize the threads we used a type of barrier that would wait for all threads to complete processing a single iteration before allowing them to continue on to the next.

Optimizations:

For optimization of the 1k test case we obviously needed to reduce the number of iterations the processing for loop performs. As GoL iterates, fewer and fewer cells become active and in fact remain in a static state for the rest of the iterations. We wanted to make use of this and instead of iterating over the entire board in a for loop every iteration, we wanted to only iterate over currently active cells and their neighbours every iteration. The number of active cells becoming smaller and smaller as the game progresses would therefore reduce our processing time if we only analyzed the active cells. To achieve this we would track changes in cells by comparing their current board value with their previous board value. If they had changed that meant they were active and would be placed in an active list, as well if that cell is active it is possible its change affected neighbours, so neighbour values would be calculated and if they had changed, they would also be added to the actives list. After an initial pass and building of the active list, our processing would now only iterate over the actives list for the remainder of the game, with the list progressively getting smaller as nodes that haven't changed would be pushed off.