Parallelisation

* The most obvious thing to parallelise is the update method since that seems to be affecting the program the most, in terms of efficiency. The way it is written is that there are two nested for loops and two nested if statements. That makes the time complexity roughly O(n2), using the worst case ***[SHOW CALCULATION]***. Grid initialisation in the constructor should be fairly efficient, but since we are initialising the grid and the updateGrid simultaneously, and each grid is a 2D array, this operation goes from simply being a linear insert with the complexity of O(n) to being about O(n2) as well ***[SHOW CALCULATION]***. As a result, they may both need some degree of parallelisation, but the update method more so than the grid initialisation. Drawing in the board is an additional point of consideration since the program has to go to each cell and check what the value of that cell is, then assign a colour to the cell, and then it draws the image based on those values. Perhaps a more efficient library could be found or maybe this is overkill.
* Things to consider:
  + Alternative drawing tools for faster processing OR parallelising the image processing manually (though it should already be fairly quick for smaller programs)

Serial program benchmarking

* On larger inputs, like 517 by 517, the output takes much longer. In this case, it took 72754 ms and 98904 steps to get to a stable state. This is much longer compared to a smaller input size of 65 by 65 which takes 29 ms and 1156 stable steps to run.