# Cellular Automaton

***USING GNU C++ COMPILER***
*Word Count: 592*

*Ramsay Sewell - 170012517 - CS*
*Maxwill Kelly - 180004073 - CS*
*Pui-Hin Vincent Lee - 180006178 - AC*

We started our creative process of developing Cellular Automata when we first received our assignment, a couple of weeks before the due date. To begin, we knew that we must start by delegating tasks and managing time efficiently for the project.

We began planning immediately, allowing ourselves a generous amount of time to work on our group planning document, which involves: each class and respective methods; an extension checklist; a list of all project files, all with specific ownership and completion status. This meant that as well as using Git and Github for versioning, we could refer to our planning document if ever in doubt of completion or whom to speak to regarding a class. For example, if we were encountering errors in a function or finding out whom to ask if we could update a variable's access definition. We believe that this document has encouraged us to communicate and therefore work brilliantly as a team.

Once we had distinguished and delegated tasks, we set off to complete them in parallel to one another. Max would investigate the storage class, which the program entirely revolves around; Ramsay would work on the Rulesets and programming the logic behind how each Ruleset produces a slightly (or extremely) different output per generation; and Vincent would step forward in the project brief, to give himself enough time to work on the Game of Life extension. We saw Vincent's time dedicated to this extension as worthwhile, as it increased everyone in the group's understanding of a Cellular Automaton, due to the complexity, as well as the potential percentage gained, with the including of this in our project.

Max speedily created his Classes for Vincent and Ramsay to work off, giving them both a great foundation. As every program contains bugs, our program was no different, however, this did not stop the three teammates from working together during Lab Sessions and some late nights in the University Library.

Once Ramsay had finished the methods to convert to and from Binary, this allowed him to set up the logic for the Rulesets. Converting a user inputted and validated number into the program, the method would convert this into its binary representation and with that, passed onto the Table class to produce each generation.

For the Game of Life, we decided to create a separate C++ file with an accompanying header file. The C++ file contained the 2 functions, one to set up the initial grid for the Game of Life itself and then Another function that can be called to run one frame of the Game of Life.

However, we did need to create functions in the premade table class to be able to run this extra correctly, for example, a function for finding the number of neighbours around a square. We were also able to add a wrapper to the program's grid, turning it into an infinite grid, improving the running of the Game of Life.

For Langton's Ant the way we display the location of the ant we're unable to show the state of the block under it due how we print chars to the console. We opted to show the location of the ant with an arrow of where it's pointing instead of prioritising showing the state of the block it's on.

In conclusion, the most challenging part of the assignment was structuring the table and using this to work with the Ruleset. With this, we believe that put in a considerable amount of effort and hard work, to complete this assignment to a good standard.