# COMP2240 Assignment 2 - Concurrency

*By Connor Farrenden (c3374676)*
*23/09/2022*

## Introduction
This report will address and discuss the implementation of various algorithms used for 3 concurrency problems that enforced mutual exclusion and are deadlock and starvation free. Particularly edge cases considered and the behaviour of each algorithm towards these, specific tricks and techniques applied, and specific issues faced.

## Edge Cases
Edge cases varied per problem. For problem 1, this involved do as little as 10 NEONs (maximum boundaries) and see the behaviour that resulted and whether output was correct. For problems 2 and 3 this involved looping through a few threads first as opposed to all the threads and working with constrained time. This also included different inputs to see whether the output would still be correct, as opposed to a fixed input set, such as varying timing inputs and ids. Trying a variety of different values for the thread sleeping was also used to identify the impact this had on times and thread management allocation, particularly as this had impacts on the output of the program when run.

## Specific Tricks/Techniques
Lots of print statements were applied throughout each the problems and the algorithms during runtime to identify the flow of the program and major issues, including general strings and the values of specific variables for each object/thread. This made it easier to identify if certain loops or conditionals were running, as well as the output of certain methods was correct in comparison to the expected output. This helped with identifying which threads were being run at specific times in the program, alongside the help with sleeping the threads, slowing it down enough to identify what thread was being at what point of the algorithm during runtime. Drawing out the flow of the algorithms was also a vital trick that helped break down each problem and ensure correct functionality.

## Specific Issues
The main issues faced across each problem was the proper management of threads to ensure correct algorithmic flow and output. A major issue with problem 2 involved threads starting and then finishing (not being accessible again), requiring looping to ensure this would not occur (holding threads in run). Another one was properly managing having more than 5 threads starting with the limited 5 permits, ensuring these were completed first before moving onto the next threads to be seated as customers in the parlour. Developing a proper sequence of locking and unlocking threads was therefore a large issue. However, many of the issues faced were fixed by sleeping the threads to allow access of resources.

## Conclusion
Overall, there was various factors involved with the implementation of each algorithm for each concurrency. Many specific issues appeared but were overcome with the use of many specific tricks and techniques and edge cases, to ensure mutual exclusion was enforced and each algorithm was deadlock and starvation free.