**Problem 1: Farmer Bridge Problem – Analysis**

The algorithm was tested against supplied and a variety of inputs to which it returned expected results. Although a queue of blocked threads is maintained when using semaphores, further measures were taken to ensure that the solution is starvation-free. A queue is maintained inside the Bridge object to assure that each farmer gets at least one turn to utilize the bridge before any other farmer does twice. If a farmer gets the lock on the bridge, then the program checks to see if it is the farmers' turn to cross the bridge. If it is the farmers' turn to cross the bridge, then the farmer is allowed to cross the bridge and the lock is released after. If it is not the farmers' turn, then the lock for the bridge is released and the farmer doesn’t cross the bridge. The queue inside the Bridge object can be thought of as a first-come-first serve or round-robin algorithm. Since all the farmers have the identical service times, then no starvation occurs due to any long process'. The algorithm prevents the occurrence of deadlocking as a consequence of a northbound and southbound farmer attempting to obtain access to the bridge at the same moment in time.

**Problem 2: Ice Cream Time Problem – Analysis**

The algorithm performs as expected and also generates expected outcomes. Other inputs were tested on the program in which all the customers arrive at the same time and it produced correct/expected outcomes. A queue of customers is maintained in the ice cream shop object in which the order of customers is stored by their arrival time. There is an external clock in the program which contains an integer value representing the time and is incremented every second until all the customer have been served/consumed their ice cream. The program conforms with the rules set out by the manager concerning serving customers when all the seats are taken. If all the seats are taken up and there is a constant stream of customers incoming, then the customers are placed inside the blocked queue in the semaphore until the shop is available to allow customers again.

**Problem 3: Hot or Iced Coffee? Problem – Analysis**

This algorithm was tested with various and provided inputs to which it performed and produced results as expected. The program doesn’t allow for more than 2 clients to use the coffee machine simultaneously and also prevents a hot and cold client from using the machine simultaneously. An ‘external clock’ is created in the algorithm which contains an integer value that is increased every second until all orders are fulfilled. I did notice that sometimes some customer tend to go before they are supposed to which is partially due to the threads operating concurrently. This was solved by making the algorithm delay the thread for a couple of microseconds at key areas in the program to ensure that the program can catch up on the actions performed.

**Relative performance of Algorithms**

All implementations versions of the algorithms perform as expected and generated the expected results. In saying that, further testing on the algorithms with various inputs can be performed to check if there are any bugs or errors in the algorithms. The external time clock object in problem B and C can have their waiting time in the time incrementing function decreased so that the algorithm performs faster and outputs are produced faster as well.