**READ ME Q1**

A deadlock can occur when there are certain combinations of circular waits such that one philosopher holds one fork along with contention for bowls and hence, all philosophers are unable to obtain required resources.

Mutex locks and condition variables are used to coordinate access to both forks and bowls in order to remove deadlocks in this code. Each philosopher attempts to acquire two forks and a bowl simultaneously instead of individually.

If any philosopher cannot acquire all the required resources, they release any acquired resources and return to a thinking state, allowing other philosophers to progress. Hence, a deadlock would never occur.

In addition to this, the code also provides a deadlock checker function that takes care of these deadlocks and notifies the user about them if they occur.

Ideally, all philosophers should have a fair opportunity to eat, but the specific frequency could differ due to the inherent randomness of the threads.