The main method used for avoiding deadlocks in my q1a and q2a programs are by acquiring locks in a given ordering. Deadlock can possibly occur if there is some form of cyclic blocking occuring (ie T0 acquires lock 0, T1 acquires lock 1, T2 acquires lock 2, T0 needs lock 1, T1 needs lock 2, and T2 needs lock 0), so by enforcing that locks of lower indices are acquired first then in the above example (and in any situation for that matter) there will always be at least one thread that can execute. (ie T0 acquires lock 0, T1 acquires lock 1, T2 needs lock 0, T0 needs lock 1, T1 acquires lock 2, and T2 needs lock 2, which implies that T1 will finish, which will free up both locks 1 and 2 allowing T0 to finish, and finally allowing T2 to finish).