i)

Deadlock occurs in the code when one thread has locked either the its ID or the BeforeID and it is trying to get the remaining one that is locked. Another thread has the ID that the first thread needs, and they come to a deadlock.

ii)

The cause of deadlock in our implementation, is that the threads are all trying to get the IDs in the same order, and eventually two or more are trying to access the same IDs and leads them to a deadlock.

iii)

Solution one

Even threads take the left ID first, odd threads take the current ID first. This would solve the problem that we are running into because there will always be one of the threads that has both of the IDs that it needs to be able to run. Once that thread is done, it will allow the next one to have the remaining ID that it requires, until all of the threads have run with their required IDs. We have not found a situation where this solution would cause a deadlock.

Solution two

Lock variables are copied into a separate object before they are locked and checks to ensure that they are both available to be locked at the same time by the same thread. This solution avoids only locking one of the required variables which would cause a deadlock. Thus, leading this solution to prevent a deadlock from happening.

Compilation Instructions