**Name: <YOUR NAME GOES HERE!!>**

**Problem Number:** Example problem (e.g., “2.A”)

Describe the ‘backstory’:

When the program starts up the main() function will create two semaphores which our two threads will share. One thread will first acquire the semaphore named ‘S’, then the semaphore named ‘Q’. The other thread will acquire the semaphores in the opposite order.

Immediately prior to the deadlock happening, the first thread successfully acquired the ‘S’ semaphore and then was suspended (as part of a normal switch from one thread to another). The second thread successfully acquired the ‘Q’ semaphore and then UNsuccessfully attempted to acquire the ‘S’ semaphore. The second thread will wait until the first thread releases the ‘S’ semaphore. The first thread eventually resumed, at which point it UNsuccessfully attempted to acquire the ‘Q’ semaphore

Brief, Clear Description of what the problem is and how it occurs:

The first thread has the ‘S’ semaphore and is waiting for the ‘Q’ semaphore.

The second thread has the ‘Q’ semaphore and is waiting for the ‘S’ semaphore.

Because both the ‘S’ and ‘Q’ semaphores can only be released by whichever thread holds it, and because the two threads are each stuck waiting for the other one to release the lock first neither thread can make process and voilà – they’re deadlocked.

|  |
| --- |
| Process #0 – the “S-then-Q” thread |
| Line # | 35 ( Q.acquire(); ) |
| Relevant Variables |
| S | This thread/process has acquired the ‘S’ semaphore, and will not release it until it also acquires the ‘Q’ semaphore |
| Q | This thread/process is waiting until it can acquire the ‘Q’ semaphore |

|  |
| --- |
| Process #1 – the “Q-then-S”thread |
| Line # | 41 ( S.acquire(); ) |
| Relevant Variables |
| Q | This thread/process has acquired the ‘Q’ semaphore, and will not release it until it also acquires the ‘S’ semaphore |
| S | This thread/process is waiting until it can acquire the ‘S’ semaphore |