The problem statement describes the n readers and 1 writer problem, which requires multiple threads to access a shared counter variable, with multiple readers able to read simultaneously but only one writer able to write at any time.

The solution to this problem can be implemented using semaphores and threads. A semaphore is a synchronization primitive that can be used to control access to shared resources. The semaphore allows multiple threads to access a shared resource simultaneously up to a specified limit. In this problem, we need to ensure that only one writer accesses the shared resource (counter variable) at a time while allowing multiple readers to read simultaneously.

To solve this problem, we need to use two semaphores - one for readers and one for writers. The reader semaphore will allow multiple readers to access the shared counter variable simultaneously, while the writer semaphore will ensure that only one writer can access the shared counter variable at any time.

In the reader code, we will use the reader semaphore to control access to the critical section. Before entering the critical section, the reader will check the writer semaphore. If the writer semaphore is set, the reader will wait until it is reset. Once the writer semaphore is reset, the reader will increment the number of readers and access the shared counter variable.

In the writer code, we will use the writer semaphore to control access to the critical section. Before entering the critical section, the writer will wait until all readers have finished accessing the shared counter variable. Once all readers have finished, the writer will set the writer semaphore, access the shared counter variable, and then reset the writer semaphore.

To ensure fairness, we will use a turnstile mechanism. The turnstile will prevent starvation of either readers or writers.