Dave Sizer
4/29/16
CS361 Programming Assignment 1 Analysis

Dekker's algorithm does not require the atomicity of statements as assumed in the textbook, because the algorithm is designed in such a way that both threads are never writing to the same resource at the same time. As proven in the book and in class, the algorithm satisfies mutual exclusion for the critical sections, so the only other part to consider is the algorithm code itself. The only variable here that is at risk of being written by both processes is turn, however the algorithm ensures that only one process sets its value at a time, since this is done right after that process' critical section. Globals are read by both processes, but this is not an issue in and of itself, since issues only arise when multiple processes attempt to modify a value based on a value that has been read.

Extra Credit:
Dekker's algorithm does not function correctly in an environment with multiple cores due to the way that cache coherency is implemented for memory shared between the cores. The cache coherency protocol does not guarantee that every core sees memory modifications in exactly the same order, which is necessary for Dekker's algorithm to function correctly.