Problem Set 1: 318

Due: Feb 23

The goal of this problem set is to help you understand how the critical section code works. Recall that the code looks like this:

Process 1:	Process 2:
S1: A = 1	S1: B = 1
S2: turn = 1	S2: turn = 2
S3: if turn!=1 goto S5	S3: if turn!=2 goto S5
S4: if B==1 goto S3	S4: if A==1 goto S3
S5:	S5:
S6: A = 0	S6: B = 0
S7: goto S1	S7: goto S1

Your job is to build a C program that simulates the behavior of the code. The global state of the code should be:

$$(A, B, turn, PC_1, PC_2).$$

The first three parts of the state are just the current values of those variables. The last two parts are the "program counters" of the two processes. Be sure to initialize the state correctly. You then need to write the routine:

void DoOneStep(int who).

This routine operates as follows: If $who == 1$ it executes *one* step for process 1; otherwise, it does the same for process 2. The key is that it only executes one single atomic step for the processes at a time.

Next write a routine that does simulate both processes at once. It should randomly decide which one to run at each step. You should print out the state so that you can watch the progress of the processes.

We then need to ask: does the system really work? For this you need to write a routine:

int IsProceesSafe().

This routine should check whether or not both processes ever are in the critical section at the same time. Then, run the simulator over and over and see if you ever get a violation to safety.

Finally, how would you use your program to check for a deadlock of the two processess?