**EE 360P HW1 Questions**

4.(15 points) Show that any of the following modifications to Peterson’s algorithm makes it incorrect:

a) A process in Peterson’s algorithm sets the turn variable to itself instead of setting it to the other process. The remaining algorithm stays the same.

P0: WantCS[0] = TRUE;

Turn = 0;

while (WantCS[1] && Turn == 1)

(switched to Process1 after P0 exits busy wait)

P1: WantCS[1] = TRUE;

Turn = 1;

while (WantCS[0] && Turn == 0)

(Then Process1 exits the busy wait)

Therefore, both P0 and P1 enter the CS.

b) A process sets the turn variable before setting the wantCS variable.

P0: Turn = 1;

(switched to P1)

P1: Turn = 0;

WantCS[1] = TRUE;

while (WantCS[0] && Turn == 0)

(Then P1 exists the busy wait to CS, since WantCS[0] is FALSE, but…)

P0: WantCS[0] = TRUE

While (WantCS[1] && Turn == 1)

(Now P0 exits the busy wait and enters CS too, because Turn was set to 0)

Therefore, both P0 and P1 have entered CS.

5.(15 points) Prove that Peterson’s algorithm is free from starvation.

After P1 has exited its CS, it will set the WantCS[1] to False; as a result, this will make the while loop for P0 false. Thus, allowing P0 to enter CS if it was attempting to while in busy wait. In addition, there is also the case that P1 will to try again to enter the CS before P0 has a chance to run. In this situation, P1 will go through its steps of setting the WantCS[1] to true and TURN to 0, but this will not allow P1 to exit busy wait since the while loop will remain true because WantCS[0] will be true. Therefore, forcing P1 to wait and will allow P0 to exit busy wait to enter CS; free of starvation.

6.(15 points) Peterson’s algorithm uses a multi-write variable turn. Modify the algorithm to use two variables turn0 and turn1 instead of turn such that P0 does not write to turn1 and P1 does not write to turn0.