Assignment 6 OS

Question 1 -

shared :
turn = 0

Algorithm for 0:
// Entry code
1. while(compare-and-swap(turn, 0, 1)) {}
<CS>
// Exit code
1. turn = 0

Algorithm for 1: (same that for 0)
// Entry code
1. while (compare-and-swap(turn, 0, 1)) {}
<CS>
// Exit code
1. turn = 0

Proof for the mutual exclusion :

Assume for the sake of contradiction that the algorithm doesn't provide mutual exclusion.
Then, both of the processes (0 and 1) are in the <CS> in the same time. Since turn is either 0 or 1, this is a contradiction to the entry code of the process 0 which lock the process until the variable turn = 0 but if the process 1 is also in the <CS> turn (by his atomicity) must be equal to 1.

Proof for the deadlock free :

Assume by contradiction that both processes 0 and 1 are stuck forever. Which mean that the turn value is not 0. Then trivially, turn = 1.
turn = 1 means that either process 0 or 1 is enter in the <CS> after changing the value of turn into 1. Then, assuming that the <CS> is finite, the turn is then changing to 0 (after a finite amount of time). This is a contradiction of the value of the variable turn.

The following scenario show that the algorithm is not starvation free :

Both of processes are stuck in the while and the variable is initialized to 0.
Process 0 get running time then entry in the process when process 1 is locked and by the way change the value of turn to 1.
After the <CS>, process 0 change the value of turn to 0.
process 0 is once again interessted and waiting in the while lock.
Repetting this scenario show that the algorithm doesn’t provide starvation free.