Distributed Algorithms CPSC-561
Assignment 1

Michael McCulloch

1 Funny Lock

Mutual exclusion is violated by two processes P and Q, by the execution order: $P_1, P_2, P_3, P_5, P_2, P_{CS}, P_7, Q_1, Q_2, Q_3, P_1, P_2, P_{CS}, Q_5, Q_2, Q_{CS}$

Deadlock-freedom is violated by two processes P and Q, by the execution order: P_1, Q_1, P_2, Q_2

2 Lousy Lock

Mutual Exclusion: Any process which enters the critical section must have it's flag with value two, and then must have checked that the other process's flag is not two. If two processes are in the critical section, then their flags must both be two, and the others may not have been two. A contradiction.

Deadlock Freedom: If > 2 processes are trying to enter the critical section, both will have $flag_i = 1$, Any process which then executes line 3 will fail, and will not set it's turn flag. Since `turn` may only hold one value, this excludes p_0 on the first invocation of the lock method, and in subsequent invocations, the last process to enter the critical section, in either case, these processes get a free pass around this loop, and will progress, satisfying deadlock freedom.

There is a special case where two processes may execute line 5 simultaneously, and so we must argue deadlock freedom for the loop at line 7 as well. The execution is as follows: $Q_1, Q_2, Q_{3.1}, P_1, P_2, P_4, Q_{3.2}, Q_2, Q_4, P_5, Q_5$. (With 3.1 being the `if` portion, and 3.2, the `then` portion of the conditional). On the second of this loop however, P will fail the test at line 3, and loop here. Allowing Q to reach line 5 alone, and make progress. Satisfying deadlock Freedom.

Starvation Freedom is not satisfied here, as if a thread sleeps before executing line 3, another thread may re-acquire the lock infinitly many times.