1. Prove the following lower bound: any no-deadlock mutual exclusion algorithm using only single-writer registers must use at least $n$ shared variables. Furthermore, prove that $n - 1$ registers must be read in the entry code.

Note that the first bound can be deduced from the theorem proved in class for multi-writer registers, but you are required to give a direct (and simpler) proof.

2. Prove that in any no-deadlock mutual exclusion algorithm using only read-write registers, the entry code for process $p$ must consists of a write to some shared variable $A$, followed by a read to a different shared variable $B$ ($B \neq A$), without $p$ writing to $B$ in between.

3. In this question we consider the algorithm for mutual exclusion using RMW operations that provides local spinning in the CC model (Algorithm 1). Modify the algorithm to provide local spinning in the DSM model, i.e., no two processes ever spin on the same variable, not even in different configurations. Justify you answer.

**Algorithm 1** Mutual exclusion using RMW operations: code for every process.

Initially $Last = 0; Flags[0] = hasLock; Flags[i] = mustWait, 0 < i < n.$

\begin{align*}
\langle \text{Entry} \rangle : \\
1: \quad & myPlace := \text{rmw}(Last, Last + 1 \mod n) \quad \triangleright \text{thread yourself on queue} \\
2: \quad & \text{wait until} \ (Flags[myPlace] = hasLock) \quad \triangleright \text{spin} \\
3: \quad & Flags[myPlace] := mustWait \quad \triangleright \text{clean} \\
\langle \text{Critical Section} \rangle \\
\langle \text{Exit} \rangle : \\
4: \quad & Flags[myPlace + 1 \mod n] := hasLock \quad \triangleright \text{tap next in line} \\
\langle \text{Remainder} \rangle
\end{align*}
4. The following is the gs object shared among two processes P and Q. Each process can perform a label operation once and a read operation many times. The sequential automaton specification of the object is as follows. It has four states, an initial one $s_0$, from which it moves to state $s_1$ if P performs a label, and from which it moves to state $s_2$ if Q performs a label. From $s_1$ it moves to $s_2$ if Q performs a label operation. From $s_2$ it moves to $s_3$ if P performs a label operation. All read operations return the current state without causing a transition.

(a) Draw the automaton of the gs object.

(b) Prove or dispute: read/write objects and gs object can implement a two-processor wait-free consensus.