# Homework Exercise #4
# Due: October 20, 2011

**4.** Consider a synchronous shared-memory system with $n$ processes. Up to $f$ of the processes can experience Byzantine failures.

The shared memory consists of a collection of read/write registers. Each register can store an integer value. Each register is owned by exactly one process. Because of the way the hardware is designed, only the owner of a register can write to that register. (It is physically impossible for any other process to write to the register, even if the process is behaving in a Byzantine manner.) Any process can read from any register. The shared memory is the only communication mechanism available to the processes.

The system is synchronous. In each round, each process can either write to one of the registers it owns or read one register. In each round, writes precede reads. Thus, if a process reads register $R$ in round $x$, the process gets the *last* value the owner of $R$ wrote during rounds 1 to $x$. (This is true even if $R$ is owned by a Byzantine process.)

Consider the Byzantine agreement problem. Each process receives an input. To solve the problem, the following three properties must be satisfied.

**Termination**: Each correct process outputs a value. (Once a correct process produces an output, it cannot produce any other output value.)

**Agreement**: No two correct processes output different values.

**Validity**: If all correct processes have input value $v$, then all correct processes must output $v$.

**(a)** Prove that it is impossible to solve Byzantine agreement in the model described above when $n = 2$ and $f = 1$.

**(b)** Give a simple algorithm for the model described above that solves Byzantine agreement for the case where inputs are binary (either 0 or 1). Be precise when you specify your algorithm: explain what each process should do in each round. For each $n$, say which values of $f$ your algorithm can handle. Prove that your algorithm is correct (for those values of $f$). For full credit, your algorithm should work for as many values of $f$ as possible.