Problem 1. Consider the safe Boolean MRSW construction shown in Figure 1. True or false: if we replace the safe Boolean SRSW register array with an array of safe $M$-valued SRSW registers, then the construction yields a safe $M$-valued MRSW register. Justify your answer.

Problem 2. You are given the algorithm in Figure 2 for constructing a single-reader single-writer (SRSW) $M$-valued atomic register using single-reader single-writer (SRSW) Boolean atomic registers. Does this proposal work? Either prove the correctness or present a counterexample.

Problem 3. Does Peterson’s two-thread mutual exclusion algorithm work if the shared atomic flag registers are replaced by regular registers?

Problem 4. Imagine running a 64-bit system on a 32-bit system, where we simulate a single 64-bit memory location (register) using two atomic 32-bit memory locations (registers). A write operation is implemented by simply writing the first 32-bits of the simulated 64-bit register in the first real register, then the second 32-bits in the second real register. A read, similarly, reads the first half from the first register, then the second half from the second register, and returns the concatenation. What is the strongest property that this 64-bit register satisfies?

- Regular register
- Safe Register
- Atomic register
- Does not satisfy any of these properties
```java
public class SafeBooleanMRSWRegister implements Register<Boolean> {
    boolean[] s_table; // array of safe SRSW registers
    public SafeBooleanMRSWRegister(int capacity) {
        s_table = new boolean[capacity];
    }

    public Boolean read() {
        return s_table[ThreadID.get()];
    }

    public void write(Boolean x) {
        for (int i = 0; i < s_table.length; i++)
            s_table[i] = x;
    }
}

public class AtomicSRSWRegister implements Register<Integer> {
    private static int RANGE = M;
    boolean[] r_bit = new boolean[RANGE]; // atomic boolean SRSW

    public AtomicSRSWRegister(int capacity) {
        for (int i = 1; i <= RANGE; i++)
            r_bit[i] = false;
        r_bit[0] = true;
    }

    public void write(int x) {
        r_bit[x] = true;
        for (int i = x - 1; i >= 0; i--)
            r_bit[i] = false;
    }

    public int read() {
        for (int i = 0; i <= RANGE; i++)
            if (r_bit[i]) {
                return i;
            }
        return -1; // impossible
    }
}
```

Figure 1: The SafeBooleanMRSWRegister class: a safe Boolean MRSW register?

Figure 2: Boolean to $M$-valued SRSW Atomic Register Algorithm