Part 1: SystemC (60 minutes)

Assume standard headers (systemc.h) and namespace (using namespace sc_core;) are included.

Problem 1.1: Counter Module (35 min)

Implement a **4-bit synchronous up-counter** with:

- Inputs: sc_in<bool> clk, sc_in<bool> reset (active-high, synchronous)
- Output: sc_out<sc_uint<4>> count
- Behavior: Resets to 0 on reset; increments on *rising edge* of clk when not resetting. Wraps at 15→0.

Tasks:

a) Complete the module skeleton below:

```
SC_MODULE(Counter4) {
  sc_in<bool> clk, reset;
  sc_out<sc_uint<4>> count;
  sc_uint<4> current_val; // Internal state

  void count_proc() {
    // YOUR CODE HERE (use if/else, no wait())
  }

  SC_CTOR(Counter4) {
    // YOUR SENSITIVITY + PORT BINDING HERE
  }
};
```

b) In 2 sentences: Why is SC_METHOD (not SC_THREAD) appropriate here?

Problem 1.2: Testbench Snippet (25 min)

Write **ONLY the simulation control block** (inside sc_main):

- Instantiate Counter4 + signals
- Generate 10ns-period clock (50% duty)
- Assert reset for first 15ns, then release
- Run simulation for **200ns total**
- Enable VCD tracing for count signal → "counter_trace.vcd"
  (No full testbench class needed—just critical setup lines)

Part 2: SystemVerilog (60 minutes)

Problem 2.1: Shift Register RTL (30 min)

Design a **parameterized 4-stage SISO shift register**:

```
module shift_reg #(
  parameter DATA_W = 8
)(
  input logic clk, reset, shift_en,
  input logic [DATA_W-1:0] data_in,
  output logic [DATA_W-1:0] data_out
);
  // YOUR RTL CODE HERE
  // Requirements:
  // • Synchronous active-high reset (clears all stages to 0)
  // • When shift_en=1: shift data_in → stage0 → stage1 → stage2 → stage3 (data_out)
  // • When shift_en=0: hold current values
Endmodule
```

Hint: Use an array of registers for internal stages.

Problem 2.2: Testbench Logic (25 min)

Complete the **stimulus block** inside an initial block:

Hint: Use an array of registers for internal stages.

Problem 2.2: Testbench Logic (25 min)

Complete the stimulus block inside an initial block:

```
initial begin
  // Reset sequence
  reset = 1; shift_en = 0; data_in = 8'hAA; #15;
  reset = 0;

  // YOUR STIMULUS SEQUENCE BELOW (total sim: 100ns):
  // • Cycles 1-4: shift_en=1, data_in increments: 8'h11 → 8'h22 → 8'h33 → 8'h44
  // • Cycles 5-6: shift_en=0 (hold state)
  // • Cycle 7: shift_en=1, data_in=8'hFF
  // • End simulation at 100ns
  // Include $dumpfile/$dumpvars setup for "shift.vcd"
End
```

(Assume clock generator runs concurrently; use # delays for timing)

Problem 2.3: Concept Check (5 min)

In **one sentence each**:
a) Why does SystemC use sc_signal instead of plain C++ variables for ports?
b) What critical error occurs if you declare data_out as wire instead of logic in Problem 2.1?