Dylan Thornburg Elen 21 prelab 7

```
module UpControl (
  input [2:0] Count,
  output reg Load,
  output reg [2:0] NewCount
);

  always @(Count) begin
    case(Count)
      3'b000:
        begin
          Load = 0;
          NewCount = 3'b001;
        end
      3'b001:
        begin
          Load = 0;
          NewCount = 3'b010;
        end
      3'b010:
        begin
          Load = 0;
          NewCount = 3'b011;
        end
      3'b011:
        begin
          Load = 0;
          NewCount = 3'b100;
        end
      3'b100:
        begin
          Load = 1;
          NewCount = 3'b001;
        end
      default:
        begin
          Load = 0;
          NewCount = 3'b000;
        end
    endcase
  end

endmodule
```

```
module DownControl (
  input [2:0] Count,
  output reg Load,
  output reg [2:0] NewCount
);

  always @(Count) begin
    case(Count)
      3'b000:
        begin
          Load = 0;
          NewCount = 3'b111;
        end
      3'b001:
        begin
          Load = 1;
          NewCount = 3'b101;
        end
      3'b010:
        begin
          Load = 0;
          NewCount = 3'b001;
        end
      3'b011:
        begin
          Load = 0;
          NewCount = 3'b010;
        end
      3'b100:
        begin
          Load = 0;
          NewCount = 3'b011;
        end
      default:
        begin
          Load = 0;
          NewCount = 3'b000;
        end
    endcase
  end
```

```
endmodule


module WinLose (
  input [2:0] UpCount,
  input [2:0] DownCount,
  input Stop,
  output reg CntEn,
  output reg Win,
  output reg Lose
);

  reg win_detected; // Internal signal to detect winning condition

  UpControl up_ctrl (
    .Count(UpCount),
    .Load(),
    .NewCount()
  );

  DownControl down_ctrl (
    .Count(DownCount),
    .Load(),
    .NewCount()
  );

  always @(UpCount, DownCount, Stop) begin
    if (Stop) begin
      CntEn = 0; // Stop asserted, disable counter
      if (UpCount == DownCount) begin
        Win = 1; // Win condition detected
        Lose = 0; // Reset lose condition
        win_detected = 1; // Remember win detection
      end
      else begin
        Win = 0; // Reset win condition
        if (!win_detected) // Only assert lose if win wasn't detected before
          Lose = 1; // Lose condition detected
      end
    end
    else begin
      CntEn = 1; // Stop not asserted, enable counter
      Win = 0; // Reset win condition
      Lose = 0; // Reset lose condition
      win_detected = 0; // Reset win detection
    end
  end

endmodule
```