```
// Eugene Ngo
// 11/28/2022
// EE 271
// Lab 5

// Winner module takes in 1-bit clk, 1-bit reset, 1-bit LED9, 1-bit LED1, 1-bit L, and 1-bit R as inputs
// and returns 7-bit HEX0, 7-bit HEX5, and 1-bit restart. This module determines who the winner of Cyber War
// is by checking to see whether the human or the computer scores and then increments the displayed score
// appropriately: if human wins then the score displayed on HEX0 is incremented and if the computer wins then
// the score on HEX5 is incremented. Once a competitor, either human or computer, wins, the game is restarted.

module winner (restart, HEX0, HEX5, clk, reset, LED9, LED1, L, R);
   input logic clk, reset;
   input logic LED9, LED1, L, R;
   output logic [6:0] HEX0, HEX5;
   output logic  restart;

   logic [2:0] Lcount;
   logic [2:0] Rcount;

   enum {off, P1, P2} ps, ns;

   always_comb begin
      case(ps)
         off:   if(LED1 & ~L & R) ns = P1;

                else if(LED9 & ~R & L) ns = P2;

                else ns = off;

         P1: ns = P1;  // Human
         P2: ns = P2;  // Computer

      endcase

   end

   // counter system

   always_comb begin

                                          // Human
         if(Rcount == 3'b000)             // 0
            HEX0 = 7'b1000000;
         else if(Rcount == 3'b001)        // 1
            HEX0 = 7'b1111001;
         else if(Rcount == 3'b010)        // 2
            HEX0 = 7'b0100100;
         else if(Rcount == 3'b011)        // 3
            HEX0 = 7'b0110000;
         else if(Rcount == 3'b100)        // 4
            HEX0 = 7'b0011001;
         else if(Rcount == 3'b101)        // 5
            HEX0 = 7'b0010010;
         else if(Rcount == 3'b110)        // 6
            HEX0 = 7'b0000010;
         else begin                       // 7
            HEX0 = 7'b1111000;
         end

                                          // Computer
         if(Lcount == 3'b000)             // 0
            HEX5 = 7'b1000000;
         else if(Lcount == 3'b001)        // 1
            HEX5 = 7'b1111001;
         else if(Lcount == 3'b010)        // 2
            HEX5 = 7'b0100100;
         else if(Lcount == 3'b011)        // 3
```

```
            HEX5 = 7'b0110000;
         else if(Lcount == 3'b100)     // 4
            HEX5 = 7'b0011001;
         else if(Lcount == 3'b101)     // 5
            HEX5 = 7'b0010010;
         else if(Lcount == 3'b110)     // 6
            HEX5 = 7'b0000010;
         else begin                    // 7
            HEX5 = 7'b1111000;
         end

   end

   always_ff @(posedge clk) begin
      if(ps == off & ns == P1) begin

         Rcount <= Rcount + 1;

      end

      else if(ps == off & ns == P2) begin

         Lcount <= Lcount + 1;
      end

      else begin
         Rcount <= Rcount;
         Lcount <= Lcount;

      end

      if(reset) begin
         Lcount <= 3'b000;
         Rcount <= 3'b000;
         ps <= off;
         restart <= 0;
      end

      if(Lcount == 3'b111) begin
         Lcount <= 3'b000;
         Rcount <= 3'b000;
         ps <= off;
         restart <= 0;
      end

      if(Rcount == 3'b111) begin
         Lcount <= 3'b000;
         Rcount <= 3'b000;
         ps <= off;
         restart <= 0;
      end

      else if(restart) begin
         ps <= off;
         restart <= 0;
      end

      else
         ps <= ns;

      if(ps == P1 | ps == P2)
         restart <= 1;
      else
         restart <= 0;
   end

endmodule

// winner_testbench tests all expected, unexpected, and edgecase behaviors
module winner_testbench();
   logic clk, reset;
   logic LED9, LED1, L, R;
   logic restart;
```

```
    logic [6:0] HEX0, HEX5;

    winner dut (.restart, .HEX0(HEX0), .HEX5(HEX5), .clk, .reset, .LED9, .LED1, .L, .R);

    parameter CLOCK_PERIOD = 100;
    initial begin
       clk <= 0;
       forever #(CLOCK_PERIOD / 2)
       clk <= ~clk;
    end

    initial begin
       reset <= 1;                                     @(posedge clk);
                                                       @(posedge clk);
       reset <= 0;                                     @(posedge clk);
                                                       @(posedge clk);
       LED9 <= 1; LED1 <= 0; L <= 1; R <= 0;  @(posedge clk);
                                                       @(posedge clk);
       LED9 <= 0; LED1 <= 1;                           @(posedge clk);
                                                       @(posedge clk);
       LED9 <= 1; LED1 <= 1;                           @(posedge clk);
                                                       @(posedge clk);
                  LED1 <= 0;          R <= 1;  @(posedge clk);
                                                       @(posedge clk);
       reset <= 1;                                     @(posedge clk);
                                                       @(posedge clk);
       reset <= 0;                                     @(posedge clk);
                                                       @(posedge clk);
       LED9 <= 0; LED1 <= 1; L <= 0; R <= 1;  @(posedge clk);
                                                       @(posedge clk);
       LED9 <= 1;                                      @(posedge clk);
                                                       @(posedge clk);
       LED9 <= 0;            L <= 1;                   @(posedge clk);
                                                       @(posedge clk);
       $stop;
    end
 endmodule
```