```
Problem 1: [12 points] Drill problem
Filename: hw5prob1.sv
AndrewID: xinyew
```

```
`default_nettype none

module DflipflopPR
  (input  logic D,
   input  logic clock, preset_L, reset_L,
   output logic Q);

  always_ff @(posedge clock, negedge reset_L, negedge preset_L)
    if (~reset_L && ~preset_L)
      Q <= 1'bx;
    else if (~reset_L)
      Q <= 1'b0;
    else if (~preset_L)
      Q <= 1'b1;
    else
      Q <= D;

endmodule : DflipflopPR

module DflipflopPR_test();

  logic clock, reset_L, preset_L, D, Q;
  DflipflopPR dut(.D(D),
                  .Q(Q),
                  .clock(clock),
                  .reset_L(reset_L),
                  .preset_L(preset_L));

  // clock wave
  initial begin
        clock = 1'b0;
    #10 clock = ~clock;
    #20 clock = ~clock;
    #20 clock = ~clock;
    #10 clock = ~clock;
    #10 clock = ~clock;
    #10 clock = ~clock;
    #10 clock = ~clock;
  end

  // reset_L wave
  initial begin
        reset_L = 1'b0;
    #5  reset_L = ~reset_L;
    #15 reset_L = ~reset_L;
    #5  reset_L = ~reset_L;
    #30 reset_L = ~reset_L;
    #5  reset_L = ~reset_L;
    #5  reset_L = ~reset_L;
    #15 reset_L = ~reset_L;
  end

  // preset_L wave
  initial begin
        preset_L = 1'b1;
    #35 preset_L = ~preset_L;
    #5  preset_L = ~preset_L;
    #45 preset_L = ~preset_L;
  end

  // D wave
  initial begin
    $monitor($time,, "D = %b, Q = %b, reset_L = %b, preset_L = %b",
              D, Q, reset_L, preset_L);
    // initial values
    D = 1'b0;
    // waveform
    #5  D = ~D;
    #10 D = ~D;
```

```
70      #5  D = ~D;
71      #5  D = ~D;
72      #10 D = ~D;
73      #10 D = ~D;
74      #10 D = ~D;
75      #20 D = ~D;
76      #10 D = ~D;
77      #10 D = ~D;
78      #5  $finish;
79    end
80 endmodule : DflipflopPR_test
```

```
Problem 2: [6 points] Drill problem
Filename: hw5prob2.sv
AndrewID: xinyew
```

```
`default_nettype none

module hw5prob2_test();
  logic value, clock, hue, reset_n, water;

  hw5prob2 dut(.hue(hue),
               .value(value),
               .clock(clock),
               .reset_n(reset_n),
               .water(water));

  initial begin
    $monitor($time,, "state: %s, nextState: %s, in: %b%b, out: %b, reset_n: %b",
             dut.state.name, dut.nextState.name, hue, value, water, reset_n);
    // init -> Red
    clock = 0;
    reset_n = 1'b0;
    reset_n <= 1'b1;

    forever #10 clock = ~clock;
  end

  initial begin
    {hue, value} <= 2'b00;
    @(posedge clock); // #10 Red + 00 -> Pink
    @(posedge clock); // #30 Pink + 00 -> Blue
    {hue, value} <= 2'b01;
    @(posedge clock); // #50 Blue + 01 -> Blue

    // Blue Reset
    #1 reset_n = 1'b0; // Blue + Reset -> Red
    #1 reset_n = 1'b1; // release reset

    @(posedge clock); // #70 Red + 01 -> Gold
    @(posedge clock); // #90 Gold + 01 -> Green
    {hue, value} <= 2'b11;
    @(posedge clock); // #110 Green + 11 -> Green
    {hue, value} <= 2'b00;
    @(posedge clock); // #130 Green + 00 -> Red
    {hue, value} <= 2'b10;
    @(posedge clock); // #150 Red + 10 -> Blue

    // Blue reset
    #1 reset_n = 1'b0; // Blue + Reset -> Red
    #1 reset_n = 1'b1; // release reset

    {hue, value} <= 2'b11;
    @(posedge clock); // #170 Red + 11 -> Pink
    @(posedge clock); // # 190 Pink + 11 -> Green

    // Green Reset
    #1 reset_n = 1'b0; // Green + Reset -> Red
    #1 reset_n = 1'b1; // release reset

    // Red Reset
    #1 reset_n = 1'b0; // Red + Reset -> Red
    #1 reset_n = 1'b1; // release reset

    @(posedge clock); // #210 Red + 11 -> Pink

    // Pink Reset
    #1 reset_n = 1'b0; // Pink + Reset -> Red
    #1 reset_n = 1'b1; // release reset

    {hue, value} <= 2'b01;
    @(posedge clock); // #230 Red + 01 -> Gold

    // Gold Reset
    #1 reset_n = 1'b0; // Gold + Reset -> Red
```

```
70      #1 reset_n = 1'b1; // release reset
71
72      #1 $finish;
73    end
74
75 endmodule : hw5prob2_test
```

```
Problem 3: [6 points] Drill problem
Filename: hw5prob3.sv
AndrewID: xinyew
```

```
`default_nettype none
module hw5prob3_test();
  logic gym, trainer, clock, reset, got_em_all;

  hw5prob3 dut(.gym(gym),
               .trainer(trainer),
               .clock(clock),
               .reset(reset),
               .got_em_all(got_em_all));

  initial begin
    $monitor($time,, "state: %s, nextState: %s, out: %b, in: %b%b",
             dut.state.name, dut.nextState.name, got_em_all, gym, trainer);
    // init
    clock = 1'b0;
    reset = 1'b1;
    reset <= 1'b0;

    forever #10 clock = ~clock;
  end

  initial begin
    {gym, trainer} <= 2'b10;
    @(posedge clock); // #10 G 10 -> E 0
    @(posedge clock); // #30 E 10 -> S 0
    @(posedge clock); // #50 S 10 -> S 1
    {gym, trainer} <= 2'b01;
    @(posedge clock); // #70 S 01 -> G 1
    @(posedge clock); // #90 G 01 -> E 0

    // E + Reset/01 -> G 1
    #1 reset = 1'b1;
    #1 reset = 1'b0;

    {gym, trainer} <= 2'b11;
    @(posedge clock); // #110 G 11 -> S 0

    // S + Reset/11 -> G 1
    #1 reset = 1'b1;
    #1 reset = 1'b0;

    {gym, trainer} <= 2'b00;
    @(posedge clock); // #130 G 00 -> S 1

    // S + Reset/00 -> G 0
    #1 reset = 1'b1;
    #1 reset = 1'b0;

    // G + Reset/00 -> G 0
    #1 reset = 1'b1;
    #1 reset = 1'b0;

    #1 $finish;
  end

endmodule : hw5prob3_test
```

Problem 6: [8 points]
Filename: hw5prob6.sv
AndrewID: xinyew

```
1  `default_nettype none
2
3  // button debouncing circuit
4  module hw5prob6
5    (input  logic button,
6     input  logic clock, reset,
7     output logic debounced);
8
9    // enum of all 14 states
10   enum logic [3:0] {
11                     SAW0 = 4'd0, STABLE1_01 = 4'd1, STABLE1_02 = 4'd2,
12                     STABLE1_03 = 4'd3, STABLE1_04 = 4'd4, STABLE1_05 = 4'd5,
13                     STABLE1_06 = 4'd6, STABLE1_07 = 4'd7, STABLE1_08 = 4'd8,
14                     STABLE1_09 = 4'd9, STABLE1_10 = 4'd10, STABLE1_11 = 4'd11,
15                     STABLE1_12 = 4'd12, FINAL_STABLE1 = 4'd13
16   } state, nextState;
17
18   // async reset FF block
19   always_ff @(posedge clock, posedge reset)
20     if (reset)
21       state <= SAW0;
22     else
23       state <= nextState;
24
25   // next state logic
26   always_comb
27     unique case (state)
28       SAW0:           nextState = (button) ? STABLE1_01 : SAW0;
29       STABLE1_01:     nextState = STABLE1_02;
30       STABLE1_02:     nextState = STABLE1_03;
31       STABLE1_03:     nextState = STABLE1_04;
32       STABLE1_04:     nextState = STABLE1_05;
33       STABLE1_05:     nextState = STABLE1_06;
34       STABLE1_06:     nextState = STABLE1_07;
35       STABLE1_07:     nextState = STABLE1_08;
36       STABLE1_08:     nextState = STABLE1_09;
37       STABLE1_09:     nextState = STABLE1_10;
38       STABLE1_10:     nextState = STABLE1_11;
39       STABLE1_11:     nextState = STABLE1_12;
40       STABLE1_12:     nextState = FINAL_STABLE1;
41       FINAL_STABLE1:  nextState = (button) ? FINAL_STABLE1 : SAW0;
42     endcase
43
44   // output logic
45   always_comb
46     case (state)
47       SAW0:    debounced = 0;
48       default: debounced = 1;
49     endcase
50
51 endmodule : hw5prob6
```