Problem 1: [8 points] Drill problem
Filename: hw6prob1.sv

```
 1 `default_nettype none
 2 module hw6prob1
 3   (input logic a, clock, reset_N,
 4    output logic found_it);
 5
 6    enum logic [2:0] {seen0 = 3'b000, seen1 = 3'b010,
 7                      seen10 = 3'b100} state, nextState;
 8
 9    always_ff @(posedge clock, negedge reset_N)
10      if(~reset_N) state <= seen0;
11      else state <= nextState;
12
13    always_comb
14      unique case (state)
15        seen0: nextState = (a) ? seen1 : seen0;
16        seen1: nextState = (a) ? seen1 : seen10;
17        seen10: nextState = (a) ? seen1: seen0;
18      endcase
19
20    always_comb begin
21      found_it = (state == seen10) & a;
22    end
23 endmodule: hw6prob1
24
25 module hw6prob1_test ();
26    logic clock, reset_N, a, found_it;
27
28    hw6prob1 fsm (.*);
29
30    initial begin
31      $monitor ($stime,, "found_it = %b, a = %b, state = %s",
32                found_it, a, fsm.state.name);
33      clock = 0;
34      reset_N = 0;
35      reset_N <= 1;
36      forever #5 clock = ~clock;
37   end
38
39   initial begin
40      a <= 1;
41      @ (posedge clock);
42      @ (posedge clock);
43      a <= 0;
44      @ (posedge clock);
45      a <= 1;
46      @ (posedge clock);
47      $finish;
48   end
49
50 endmodule: hw6prob1_test
51
```

Problem 2: [8 points] Drill problem
Filename: hw6prob2.sv

```
 1 `default_nettype none
 2 module hw6prob2
 3   (input logic b, clock, reset,
 4    output logic found3zeros_N);
 5
 6    enum logic [2:0] {seen1 = 3'b000, seen0 = 3'b001, int1 = 3'b010,
 7                      int2 = 3'b011, seen00 = 3'b100, seen000 = 3'b101}
 8                      state, nextState;
 9
10    always_ff @ (posedge clock, posedge reset)
11      if (reset) state <= seen1;
12      else state <= nextState;
13
14    always_comb
15      unique case (state)
16        seen1: nextState = (b) ? seen1 : seen0;
17        seen0: nextState = (b) ? int1 : seen00;
18        int1: nextState = (~b) ? seen00 : int2;
19        int2: nextState = (~b) ? seen00 : seen1;
20        seen00: nextState = (~b) ? seen000 : int1;
21        seen000 : nextState = (~b) ? seen000 : seen00;
22      endcase
23
24    always_comb begin
25      found3zeros_N = (state != seen000);
26    end
27 endmodule: hw6prob2
28
29 module hw6prob2_test();
30   logic b, clock, reset, found3zeros_N;
31   hw6prob2 fsm (.b, .clock, .reset, .found3zeros_N);
32
33   initial begin
34     $monitor ($stime,, "b = %b, state = %s, found3zeros_N = %b",
35               b, fsm.state.name, found3zeros_N);
36     clock = 0;
37     reset = 1;
38     reset <= 0;
39     forever #5 clock = ~clock;
40   end
41
42   initial begin
43     b <= 0;
44     @ (posedge clock);
45     @ (posedge clock);
46     @ (posedge clock);
47     @ (posedge clock);
48     b <= 1;
49     @ (posedge clock);
50     @ (posedge clock);
51     @ (posedge clock);
52     @ (posedge clock);
53     @ (posedge clock);
54     $finish;
55   end
56 endmodule: hw6prob2_test
57
58
59
60
61
62
63
64
65
66
67
68
69
70
```

Problem 3: [8 points] Drill problem
Filename: hw6prob3.sv

```
 1 `default_nettype none
 2 module hw6prob3
 3   (input logic A, B, clock, reset_N,
 4    output logic F);
 5
 6    enum logic [8:0] {idle = 9'b000000001, aA = 9'b000000010,
 7                      AA = 9'b000000100, AAA = 9'b000001000,
 8                      nA = 9'b000010000, bB = 9'b000100000,
 9                      BB = 9'b001000000, BBB = 9'b010000000,
10                      nB = 9'b100000000} state, nextState;
11    always_ff @ (posedge clock, negedge reset_N)
12      if (~reset_N) state <= idle;
13      else state <= nextState;
14
15    always_comb
16      unique case (state)
17        idle: begin
18                if (A) nextState = aA;
19                else if (B) nextState = bB;
20                else nextState = idle;
21              end
22        aA: begin
23                if (A) nextState = AA;
24                else nextState = nA;
25           end
26        AA: begin
27                if (A) nextState = AAA;
28                else nextState = nA;
29           end
30        AAA: begin
31                if (A) nextState = aA;
32                else if (B) nextState = bB;
33                else nextState = idle;
34             end
35        nA: begin
36                if (A) nextState = aA;
37                else nextState = nA;
38            end
39        bB: begin
40                if (B) nextState = BB;
41                else nextState = nB;
42           end
43        BB: begin
44                if (B) nextState = BBB;
45                else nextState = nB;
46            end
47        BBB: begin
48                if (A) nextState = aA;
49                else if (B) nextState = bB;
50                else nextState = idle;
51             end
52        nB: begin
53                if (B) nextState = bB;
54                else nextState = nB;
55            end
56      endcase
57
58    always_comb begin
59      F = ((state == AA) & A) | ((state == BB) & B);
60    end
61
62 endmodule: hw6prob3
63
64
```

Problem 4: [4 points] Drill problem
Filename: hw6prob4.sv

```
 1 `default_nettype none
 2
 3 module hw6prob4 ();
 4    logic Input, Prob1, clock, reset_n;
 5
 6    hw5prob1 dut (.Input, .clock, .reset_n, .Prob1);
 7
 8    initial begin
 9      $monitor ($stime,, "Prob1 = %b, Input = %b, state = %s",
10                Prob1, Input, dut.state.name);
11      clock = 0;
12      reset_n = 0;
13      reset_n <= 1;
14      forever #5 clock = ~clock;
15   end
16
17   initial begin
18      Input <= 1;
19      @ (posedge clock);
20      Input <= 0;
21      @ (posedge clock);
22      @ (posedge clock);
23      Input <= 1;
24      @ (posedge clock);
25      Input <= 0;
26      @ (posedge clock);
27      @ (posedge clock);
28      Input <= 1;
29      @ (posedge clock);
30      Input <= 0;
31      @ (posedge clock);
32      Input <= 1;
33      @ (posedge clock);
34      @ (posedge clock);
35      $finish;
36   end
37
38 endmodule: hw6prob4
39
```

Problem 5: [4 points] Drill problem
Filename: hw6prob5.sv

```
 1 `default_nettype none
 2
 3 module hw6prob5 ();
 4    logic Input, Prob2, clock, reset_n;
 5
 6    hw5prob2 dut (.Input, .clock, .reset_n, .Prob2);
 7
 8    initial begin
 9      $monitor ($stime,, "Prob2 = %b, Input = %b, state = %s",
10                Prob2, Input, dut.state.name);
11      clock = 0;
12      reset_n = 0;
13      reset_n <= 1;
14      forever #5 clock = ~clock;
15   end
16
17   initial begin
18      Input <= 1;
19      @ (posedge clock);
20      Input <= 0;
21      @ (posedge clock);
22      @ (posedge clock);
23      Input <= 1;
24      @ (posedge clock);
25      Input <= 0;
26      @ (posedge clock);
27      Input <= 1;
28      @ (posedge clock);
29      @ (posedge clock);
30      $finish;
31   end
32
33 endmodule: hw6prob5
34
```

Problem 6: [16 points]
Filename: hw6prob6.sv

```
 1 `default_nettype none
 2 module hw6prob6
 3   (input logic clock, reset, r1, r2, r3,
 4    output logic g1, g2, g3);
 5
 6    enum logic [2:0] {idle = 3'b000, A = 3'b001,
 7                      B = 3'b010, C = 3'b100} state, nextState;
 8
 9    always_ff @(posedge clock, posedge reset)
10      if(reset) state <= idle;
11      else state <= nextState;
12
13    always_comb begin
14      if ((g1 & r1) | (r1 & (~r2) & (~r3)) |
15          (r1 & (~g1) & (~g2) & (~g3)) |
16          ((~g2) & r1 & r2 & (~r3))) nextState = C;
17      else if (((~g3) & (~r1) & (r2)) | (g3 & (~r1) & (r2) & (~r3)) | (g2 & r2))
18          nextState = B;
19      else if (((~g3) & (~r1) & (~r2) & r3) | ((~g1) & (~g2) & g3 & r3))
20          nextState = A;
21      else
22          nextState = idle;
23    end
24
25    always_comb begin
26      g1 = state[2];
27      g2 = state[1];
28      g3 = state[0];
29    end
30 endmodule: hw6prob6
31
32 module hw6prob6_test1 ();
33    logic clock, reset, r1, r2, r3, g1, g2, g3;
34
35    hw6prob6 fsm (.*);
36
37    initial begin
38      $monitor ($stime,, "r1 = %b, r2 = %b, r3 = %b, \
39                g1 = %b, g2 = %b, g3 = %b, state = %s",
40                r1, r2, r3, g1, g2, g3, fsm.state.name);
41      clock = 0;
42      reset = 1;
43      reset <= 0;
44      forever #5 clock = ~clock;
45   end
46
47   initial begin
48      r1 = 0;
49      r2 = 0;
50      r3 = 0;
51      @ (posedge clock);
52      r2 = 1;
53      r3 = 1;
54      @ (posedge clock);
55      @ (posedge clock);
56      r2 = 0;
57      r1 = 1;
58      @ (posedge clock);
59      @ (posedge clock);
60      r1 = 0;
61      @ (posedge clock);
62      @ (posedge clock);
63      r3 = 0;
64      @ (posedge clock);
65     $finish;
66   end
67
68 endmodule: hw6prob6_test1
69
70 module hw6prob6_test2 ();
```

```
 71    logic clock, reset, r1, r2, r3, g1, g2, g3;
 72
 73    hw6prob6 fsm (.*);
 74
 75    initial begin
 76      $monitor ($stime,, "r1 = %b, r2 = %b, r3 = %b, \
 77                g1 = %b, g2 = %b, g3 = %b, state = %s",
 78                r1, r2, r3, g1, g2, g3, fsm.state.name);
 79      clock = 0;
 80      reset = 1;
 81      reset <= 0;
 82      forever #5 clock = ~clock;
 83   end
 84
 85   initial begin
 86      r1 = 0;
 87      r2 = 0;
 88      r3 = 0;
 89      @ (posedge clock);
 90      r3 = 1;
 91      @ (posedge clock);
 92      r3 = 0;
 93      r2 = 1;
 94      @ (posedge clock);
 95      r2 = 0;
 96      r1 = 1;
 97      @ (posedge clock);
 98      r1 = 0;
 99      @ (posedge clock);
100      r3 = 1;
101      @ (posedge clock);
102      r2 = 1;
103      @ (posedge clock);
104      r2 = 0;
105      r1 = 1;
106      @ (posedge clock);
107      r2 = 2;
108      @ (posedge clock);
109      r3 = 0;
110      r2 = 1;
111      r1 = 0;
112      @ (posedge clock);
113      r3 = 1;
114      @ (posedge clock);
115      r3 = 0;
116      r1 = 1;
117      @ (posedge clock);
118      r1 = 0;
119      r3 = 1;
120      @ (posedge clock);
121      r1 = 1;
122      @ (posedge clock);
123      r3 = 0;
124      r1 = 1;
125      r2 = 0;
126      @ (posedge clock);
127      r3 = 1;
128      @ (posedge clock);
129      r3 = 0;
130      r2 = 1;
131      @ (posedge clock);
132      r3 = 1;
133      @ (posedge clock);
134      $finish;
135   end
136
137 endmodule: hw6prob6_test2
138
139
```