```
1   //
2   // Digital Logic Lab Part 2
3   // Neil Nie
4   // (c) Yongyang Nie
5   // Oct 30th, 2018
6   //
7
8   // this only displays nothing or 1.
9
10  module half_display(
11
12     in,          // accepts a four bit input
13     out          // returns a six bit output
14  );
15
16  input in;          // A
17  output [6:0] out; // D C B A G
18
19
20  assign out[6] = 1;
21  assign out[5] = 1;
22  assign out[4] = 1;
23  assign out[3] = 1;
24  assign out[2] = ~in;
25  assign out[1] = ~in;
26  assign out[0] = 1;
27
28  endmodule
29
30  // full seven segment display module.
31
32  module seven_segment_display(
33
34     in,          // accepts a four bit input
35     out          // returns a six bit output
36  );
37
38  input [3:0] in;   // A B C D
39  output [6:0] out; // F E D C B A
40
41  assign out[6] = (~in[3] & ~in[2] & ~in[1]) + (~in[3] & in[2] & in[1] & in[0]);
42  assign out[5] = (in[3] & in[2] & ~in[1]) + (~in[3] & in[1] & in[0]) + (~in[3] & ~in[2] & in[
    0]) + (~in[3] & ~in[2] & in[1]);
43  assign out[4] = (~in[1] & in[2]) | (in[0]);
44  assign out[3] = (in[2] & in[1] & in[0]) + (~in[3] & in[2] & ~in[1] & ~in[0]) + (in[3] & ~in[
    2] & in[1] & ~in[0]) + (~in[2] & ~in[1] & in[0]);
45  assign out[2] = (in[3] & in[2] & ~in[0]) + (~in[3] & ~in[2] & in[1] & ~in[0]) + (in[3] & in[
    2] & in[1]);
46  assign out[1] = (in[2] & in[1] & ~in[0]) + (in[3] & in[1] & in[0]) + (~in[3] & in[2] & ~in[1
    ] & in[0]) + (in[3] & in[2] & ~in[0]);
47  assign out[0] = (in[2] & ~in[1] & ~in[0]) + (in[3] & in[2] & ~in[1]) + (~in[3] & ~in[2] & ~
    in[1] & in[0]) + (in[3] & ~in[2] & in[1] & in[0]);
48
49  endmodule
50
51  module two_one_four_bit_mux(
52
53     a,
54     b,
55     sel,
56     out
57  );
58
59  input [3:0] a;
60  input [3:0] b;
61  output [3:0] out;
62  input sel;
63
64  assign out[3] = (~sel & a[3]) | (sel & b[3]);
65  assign out[2] = (~sel & a[2]) | (sel & b[2]);
66  assign out[1] = (~sel & a[1]) | (sel & b[1]);
67  assign out[0] = (~sel & a[0]) | (sel & b[0]);
68
69  endmodule
70
71  // returns 1 if in > 9
```

```
72   // returns 0 if in < 9
73
74   module comparator(
75
76      in,
77      out
78
79   );
80
81   input [3:0] in;
82   output out;
83
84   assign out = in[3] & (in[2] | in[1]);
85
86   endmodule
87
88   // this converter module convert number > 9
89   // to the appropriate number
90
91   module converter(
92
93      in,
94      out
95   );
96
97   input [3:0] in;
98   output [3:0] out;
99
100  assign out[3] = 0;
101  assign out[2] = in[3] & in[2] & in[1];
102  assign out[1] = in[3] & in[2] & ~in[1];
103  assign out[0] = in[3] & in[0];
104
105  endmodule
106
107  module Part2(
108
109     SW,
110     HEX2,
111     HEX3
112  );
113
114  input [3:0] SW;
115  output [6:0] HEX2;
116  output [6:0] HEX3;
117
118  wire compare_out;
119  wire [3:0] convert_out;
120  wire [3:0] mux_out;
121  wire [6:0] display1;
122  wire [6:0] display0;
123
124  comparator(SW, compare_out);
125  converter(SW, convert_out);
126
127  two_one_four_bit_mux (SW, convert_out, compare_out, mux_out);
128  seven_segment_display (mux_out, display0);
129  half_display(compare_out, display1);
130
131  assign HEX2 = display0;
132  assign HEX3 = display1;
133
134  endmodule
135
136
137
```