```
1   /* Name: Eugene Ngo
2      Date: 1/13/2023
3      Class: EE 371
4      Lab 1: Parking Lot Occupancy Counter*/
5
6   // carCount takes two inputs (inc, dec). It adds 5'b00001 to out when inc is true, and
    subtracts 5'b00001
7   // from out when dec is true. Out has a minimum value of 5'b00000 and a maximum value
    determined by the
8   // parameter (25 by default).
9   module carCount #(parameter MAX=25) (inc, dec, out, full, clear, clk, reset);
10
11     input logic inc, dec, clk, reset;
12     output logic [4:0] out;
13     output logic full, clear;
14
15     // Sequential logic for counting up and counting down depending on the input.
16     always_ff @(posedge clk) begin
17        if (reset) begin
18           out <= 5'b00000;
19           full <= 1'b0;
20           clear <= 1'b0;
21        end
22        else if (inc & out < MAX) begin //increment when not at max
23           out <= out + 5'b00001;
24           clear <= 1'b0;
25        end
26        else if (dec & out > 5'b00000) begin // decrement when not at min
27           out <= out - 5'b00001;
28           full <= 1'b0;
29        end
30        else if (out == MAX) begin // hold value at max, output full
31           out <= MAX;
32           full <= 1'b1;
33        end
34        else if (out == 5'b00000) begin // hold value at min, output clear
35           out <= 5'b00000;
36           clear <= 1'b1;
37        end
38        else
39           out <= out; // hold value otherwise
40     end // always_ff
41
42  endmodule
43
44  // carCount_testbench tests all expected, unexpected, and edgecase behaviors
45  module carCount_testbench();
46     logic inc, dec, full, clear, reset;
47     logic [4:0] out;
48     logic CLOCK_50;
49
50     carCount #(5) dut (.inc, .dec, .out, .full, .clear, .clk(CLOCK_50), .reset);
51
52     // Setting up the clock.
53     parameter CLOCK_PERIOD = 100;
54     initial begin
55        CLOCK_50 <= 0;
56        forever #(CLOCK_PERIOD/2) CLOCK_50 <= ~CLOCK_50; // toggle the clock forever
57     end // initial
58
59     initial begin
60        reset <= 1;                          repeat(3) @(posedge CLOCK_50); // reset
61        reset <= 0; inc <= 1;                repeat (7) @(posedge CLOCK_50); // inc past max limit
62                    inc <= 0; dec <= 1; repeat(7) @(posedge CLOCK_50); // dec past min limit
63        $stop;
64     end
65  endmodule // counter_testbench
```