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