```
module up_counter (clk, reset, enter, exit, counter);

   input logic clk, reset;
   input logic enter, exit;
   output logic [4:0] counter;

   logic [4:0] counter_up;

   always_ff @(posedge clk) begin
      if (reset) counter_up <= 5'b00000;
      else if (enter) counter_up <= counter_up + 5'd00001;
      else if (exit) counter_up <= counter_up - 5'd00001;
      else counter_up <= counter_up;
   end

   assign counter = counter_up;

endmodule

module up_counter_testbench();

   logic clk, reset;
   logic enter, exit;
   logic [4:0] counter;

   up_counter dut (.clk, .reset, .enter, .exit, .counter);

   parameter CLOCK_PERIOD = 100;
   initial clk = 1;
   always begin
      #(CLOCK_PERIOD / 2);
      clk = ~clk;
   end

   initial begin
                  @(posedge clk);
   reset <= 1;    @(posedge clk);
                  @(posedge clk);
   reset <= 0;    @(posedge clk);
                  @(posedge clk);
         enter = 0; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 0;           @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 1; exit = 0; @(posedge clk);
         enter = 0; exit = 1; @(posedge clk);
         enter = 0; exit = 1; @(posedge clk);
         enter = 0; exit = 1; @(posedge clk);
         enter = 0; exit = 1; @(posedge clk);
         enter = 0; exit = 1; @(posedge clk);
         enter = 0; exit = 1; @(posedge clk);
         enter = 0; exit = 1; @(posedge clk);
```

```
76              enter = 0; exit = 1; @(posedge clk);
77              enter = 0; exit = 1; @(posedge clk);
78              enter = 0; exit = 1; @(posedge clk);
79              enter = 0; exit = 1; @(posedge clk);
80              enter = 0; exit = 1; @(posedge clk);
81              enter = 0; exit = 1; @(posedge clk);
82              enter = 0; exit = 1; @(posedge clk);
83              enter = 0; exit = 1; @(posedge clk);
84              enter = 0; exit = 1; @(posedge clk);
85              enter = 0; exit = 1; @(posedge clk);
86              enter = 0; exit = 1; @(posedge clk);
87              enter = 0; exit = 1; @(posedge clk);
88              enter = 0; exit = 1; @(posedge clk);
89              enter = 0; exit = 1; @(posedge clk);
90              enter = 0; exit = 1; @(posedge clk);
91              enter = 0; exit = 1; @(posedge clk);
92              enter = 0; exit = 1; @(posedge clk);
93              enter = 0; exit = 1; @(posedge clk);
94              enter = 0; exit = 1; @(posedge clk);
95          $stop;
96      end
97
98  endmodule
99
```