```
module normalLight (clk, reset, L, R, NL, NR, lightOn);
    input logic clk, reset;
    // L is true when left key is pressed, R is true when the right key
    // is pressed, NL is true when the light on the left is on, and NR
    // is true when the light on the right is on.
    input logic L, R, NL, NR;
    // when lightOn is true, the normal light should be on.
    output logic lightOn;
    // Your code goes here!!

    enum {on, off} ps, ns;

    always_comb begin
        case(ps)
            on:   if(~R & L | R & ~L) ns = off;
                      else ns = on;

            off:      if(NR & L & ~R | NL & R & ~L) ns = on;
                      else ns = off;

        endcase
    end

    always_comb begin
        case(ps)
            on: lightOn = 1;

            off: lightOn = 0;
        endcase
    end

    always_ff @(posedge clk) begin
        if(reset)
            ps <= off;
        else
            ps <= ns;
    end
endmodule

module normalLight_testbench();
    logic clk, reset, L, R, NL, NR, lightOn;

    normalLight dut (.clk, .reset, .L, .R, .NL, .NR, .lightOn);

    parameter CLOCK_PERIOD = 100;
    initial begin
        clk <= 0;
        forever #(CLOCK_PERIOD / 2)
        clk <= ~clk;
    end

    initial begin
        reset <= 1; @(posedge clk);
        reset <= 0; @(posedge clk);
        L <= 0; R <= 0; NL <= 0; NR <= 0; repeat(4) @(posedge clk);
                                          NR <= 1; repeat(4) @(posedge clk);
                                NL <= 1; NR <= 0; repeat(4) @(posedge clk);
                                          NR <= 1; repeat(4) @(posedge clk);
                    R <= 1; NL <= 0; NR <= 0; repeat(4) @(posedge clk);
                                          NR <= 1; repeat(4) @(posedge clk);
                                NL <= 1; NR <= 0; repeat(4) @(posedge clk);
                                          NR <= 1; repeat(4) @(posedge clk);
        L <= 1; R <= 0; NL <= 0; NR <= 0; repeat(4) @(posedge clk);
                                          NR <= 1; repeat(4) @(posedge clk);
                                NL <= 1; NR <= 0; repeat(4) @(posedge clk);
                                          NR <= 1; repeat(4) @(posedge clk);
                    R <= 1; NL <= 0; NR <= 0; repeat(4) @(posedge clk);
                                          NR <= 1; repeat(4) @(posedge clk);
                                NL <= 1; NR <= 0; repeat(4) @(posedge clk);
```

```
70                                          NR <= 1; repeat(4) @(posedge clk);
71          $stop;
72      end
73
74  endmodule
```