(3)

```
module Exam(sm_in, sm_clock, sm_out, reset);          (1)
parameter idle = 2'b00;
parameter read = 2'b01;
parameter write = 2'b11;
parameter wait = 2'b10;                               (2)
input sm_clock, reset, sm_in;
output sm_out;
reg [1:0] current_state, next_state;
always @(posedge sm_clock)
begin
if (reset == 1'b1)                                    (3)
current_state <= 2'b00;
else
current_state <= next_state;                          (4)
end
always @(current_state or sm_in)
begin
 sm_out = 1'b1;
 next_state = current_state;
 case current_state
 idle:
  sm_out = 1'b0;
  if (sm_in)
  next state = 2'b11;
  write:
  sm_out = 1'b0;
  if sm_in == 1'b0
  next_state = 2'b10;
  read:
  if (sm_in == 1'b1)
  next_state = 2'b01;
  wait:                                               (5)
  if (sm_in == 1'b1)
  next_state = 2'b00;
  endcase
  end
  endmodule
```

(4)