2 Verilog [20 points]

Please answer the following three questions about Verilog.

(a) [5 points] Does the following code result in a single D Flip-Flop with a synchronous active-low reset? Please explain your answer.

```
1  module mem (input clk, input reset, input [1:0] d, output reg [1:0] q);
2  always @ (posedge clk or negedge reset)
3      begin
4        if (!reset) q <= 0;
5        else q <= d;
6      end
7  endmodule
```

No.
The code implements *two* D Flip-Flops, not *one*. Each D Flip-Flop works with an *asynchronous* active-low reset signal.

Explanation:

- D and Q signals are two-bit-wide. Therefore, this code implements two D flip-flops.
- The reset input is included in the sensitivity list, therefore it is not synchronous.
- The code resets the output if the reset signal is low. Thus, the reset signal is active-low.

(b) [5 points] Does the following code result in a sequential circuit or a combinational circuit? Please explain your answer.

```
1  module Mask (input [1:0] data_in, input mask, output reg [1:0] data_out);
2  always @ (*)
3      begin
4          data_out[1] = data_in[1];
5          if (mask)
6              data_out[0] = 0;
7      end
8  endmodule
```

Sequential circuit.

Explanation:
This code results in a sequential circuit, as all the left-hand side signals are not assigned in every possible condition. For example, `data_out[0]` is not assigned when mask signal equals to zero.

(c) [10 points] Is the following code syntactically correct? If not, please explain the mistake(s) and how to fix it/them.

```
1   module fulladd(input a, b, c, output reg s, c_out);
2      assign s = a^b;
3      assign c_out = (a & b) | (b & c) & (c & a);
4   endmodule
5
6   module top ( input wire [5:0] instr, input wire op, output z);
7
8     reg[1:0] r1, r2;
9     wire [3:0] w1, w2;
10
11    fulladd FA1 (.a(instr[0]), .b(instr[1]), .c(instr[2]),
12                               .c_out(r1[1]), .z(r1[0]));
13    fulladd FA2 (.a(instr[3]), .b(instr[4]), .c(instr[5]),
14                               .z(r2[0]), .c_out(r2[1]));
15
16    assign z = r1 | op;
17    assign w1 = r1 + 1;
18    assign w2 = r2 << 1;
19    assign op = r1 ^ r2;
20
21  endmodule
```

The code is *not* syntactically correct.

Explanation:

- 'r1' and 'r2' have to be declared as *wires*.
- 'op' signal is connected to multiple drivers. It gets assigned from the input port and in line 19.
- The module 'fulladd' does not have ports named 'z'. Those need to be changed to 's'.
- The output signals 's' and 'c_out' have to be declared as *wires* but not as *regs*, since they are driven by *assign* statements.