1. Determine which lines have syntax errors in the accompanying Verilog code. There are at least 10 errors

10 / 10 points

```
module find_errors                    // line 1
  input  a[0:3];                      // line 2
  output [3:0]b;                      // line 3
  input [5:0]c                        // line 4
                                      // line 5
  wire [0:3]aw;                       // line 6
  wire [3:0]bw;                       // line 7
  reg [5:0]creg                       // line 8
begin                                 // line 9
  assign aw = a;                      // line 10
  assign b = bw;                      // line 11
  assign creg = c;                    // line 12
always                                // line 13
  begin                               // line 14
    if (creg = 4'h F)                 // line 15
       bw <= aw;                      // line 16
    else                              // line 17
     bw <= '0101';                    // line 18
    end;                              // line 19
  end process;                        // line 20
end                                   // line 21
```

Select only the lines below which have errors in the code listed above.

- [x] Error in line 1

  Correct
  no start paren

- [x] Error in line 2

  Correct
  dimensions in wrong place, semi-colon instead of comma

- [x] Error in line 3

  Correct
  semi-colon instead of comma

- [x] Error in line 4

  Correct
  no end paren or semi-colon

- [ ] Error in line 5
- [ ] Error in line 6
- [ ] Error in line 7
- [x] Error in line 8

  Correct
  missing semi-colon

- [x] Error in line 9

  Correct

- [ ] Error in line 10
- [ ] Error in line 11
- [ ] Error in line 12
- [x] Error in line 13

Correct
missing sensitivity list

- [ ] Error in line 14
- [x] Error in line 15

Correct
incorrect logic test (= instead of ==), array size mismatch

- [x] Error in line 16

Correct
cannot assign to a wire inside of always

- [ ] Error in line 17
- [x] Error in line 18

Correct
cannot assign to a wire inside of always, bad number format

- [x] Error in line 19

Correct
extra semi-colon

- [x] Error in line 20

Correct
no end needed, extra

- [x] Error in line 21

Correct
needs to be endmodule

Congratulations! You passed!

Grade received 87.50% Latest Submission Grade 87.50% To pass 80% or higher

Go to next item

1. Which of the following methods is a Verilog modeling styles? (**Mark all that apply**) **1 / 1 point**

- [x] Structural (gate-level)

  Correct
  Correct.

- [ ] instantiated (instance units)
- [x] behavioral (always)

  Correct

- [x] dataflow (assign)

  Correct

2. Which of the following statements is correct? (**Mark all that apply**) **1 / 1 point**

- [ ] Combinatorial logic circuits require a clock edge to operate?
- [x] "Net" data type must be driven continuously.

Correct
Correct.

- [ ] "Register" datatypes includes "tri" and "reg" .
- [x] 'Nets" datatypes are used to wire up instantiations.

Correct
Correct.

3. Which of the following statements is correct in Verilog? (**Mark all that apply**) **0.5 / 1 point**

- [ ] Integer datatype represents general-purpose variables.
- [ ] Implicitly declared "reg" types can store unsigned numbers.
- [x] "reg" can be modeled as a wire or as a storage.

Correct
Correct, depending on the context it can.

- [x] "reg" is a short form for 'register'

This should not be selected
Incorrect, please review the slides

4. Verilog supports the following logic values: x, z, 0, 1. An FPGA I/O will measure the following values by a voltmeter. High is 2.5V: **1 / 1 point**

- [x] Logic value 0: Voltmeter 0.0V.

Correct
Yes.

- [ ] Logic value x: Voltmeter 2.5V.
- [x] Logic value 1: Voltmeter 2.5V.

Correct
Yes, the voltage will be 2.5V for a 2.5V IO.

- [ ] Logic value z: Voltmeter 2.5V.

5. Blocking assignments in verilog (=) execute in series in an always block. **1 / 1 point**

- ( ) = assignments operate in parallel
- (•) = assignments operate in series.

Correct
Correct, they operate in series inside an always block.

6. If Z_out = 3'b101 the using the replication operator {2{Z_out}} creates: **1 / 1 point**

- ○ 1010
- ◉ 101101

Correct
Yes.

7. Which of the following statements is correct in Verilog? (**Mark all that apply**) **0.5 / 1 point**

- ☑ Combinatorial circuits should include all inputs for the circuit in the sensitivity list:

Correct
Yes, all inputs should be included.

- ☐ "assign" statement must be used in a sequential block (begin … end)
- ☑ When using port names in module instantiation, one shall be careful about order of the ports.

This should not be selected
No need to follow port ordered lists when using port names

- ☐ Blocking or non-blocking assignments can be used in sequential blocks.

8. Select all correct statements: **1 / 1 point**

- [ ] Since verilog is not case sensitive, the following statements are equivalent:

  C_IN = A;

  c_in = A;

- [x] "Time" is a datatype and is not supported for synthesis.

  Correct
  Correct, it's used for simulation

- [x] "===" is the case equality operator and is not synthesizable

  Correct
  This is correct

- [ ] The following assignments are equal:

  assign y = (a | b) & ~c;

  assign y = a | b & ~c;

9. Select all the correct answers from the following statements: **0.75 / 1 point**

- [ ] Verilog uses a positional or ordered port list for instances, so the following ordered port lists are equivalent:

9. Select all the correct answers from the following statements: **0.75 / 1 point**

- [ ] Verilog uses a positional or ordered port list for instances, so the following ordered port lists are equivalent:

  add4 unit_1 (a, b, c_in, c_out, sum);

  add4 unit_2 (a, b, c_out, c_in, sum);

- [x] supply0 and supply1 are data types representing ground and power respectively

  Correct
  Correct, these are wires tied to logic 0 and 1.

- [x] When connecting modules, inputs can be Nets or Registers, outputs must be Nets (wire, etc.)

  This should not be selected
  No, please refer to the slides and reading material.

- [x] The left-hand-side (LHS) of procedural assignments must be of a Register type.

  Correct
  Correct, as mentioned in slides.

10. reg [31:0] my_data; **1 / 1 point**

assign my_data = 16'hCAFE;

The reg my_data has the 32 bit value as:

Correct, these are wires tied to logic 0 and 1.

- [x] When connecting modules, inputs can be Nets or Registers, outputs must be Nets (wire, etc.)

This should not be selected
No, please refer to the slides and reading material.

- [x] The left-hand-side (LHS) of procedural assignments must be of a Register type.

Correct
Correct, as mentioned in slides.

10. reg [31:0] my_data; **1 / 1 point**

assign my_data = 16'hCAFE;

The reg my_data has the 32 bit value as:

- (•) my_data = 0000_CAFE
- ( ) my_data = FFFF_CAFE
- ( ) my_data = XXXX_CAFE

Correct
Yes, the other bits fill in with zeros.