1. Describe the following delays with an example:
    a. Regular assignment delay
       Example: #5 a = b;
       This introduces a delay of 5 time units before assigning the value of b to a.
    b. Implicit continuous assignment delay
       Example: assign #5 a = b;
       In this case, the value of b is assigned to a after a delay of 5 units continuously. Any change in b will reflect in a after 5 units of delay.
    c. Net declaration delay
       Example: wire #5 a = b;
       This introduces a delay when driving a wire, so any change in b will reflect in a after 5 time units.
2. What is delta simulation time?
   Delta time refers to an infinitesimally small time step the simulator uses to process events that logically occur simultaneously. It helps in resolving combinational logic updates within the same simulation timestep.
3. Explain transport delay inertial delay.
    - **Transport Delay:** Transmits the signal with the specified delay, regardless of glitches or short pulses.
      Example: #5 a = b;
    - **Inertial Delay:** Ignores input pulses that are shorter than the delay period.
      Example: #5 a <= b;
4. What is meant by inferring latches and how to avoid it?
   Inferring latches happen when all possible output conditions are not defined in a combinational block, leading to the memory of previous values (a latch).
   To avoid them make sure all branches (e.g., all if/else or case statements) assign values to the output signal.
5. Explain the repeat loop.
   This loop repeats the block of code a specified number of times.
6. Why is it necessary to list all inputs in the sensitivity list of a combinational circuit?
   Missing inputs in the sensitivity list can lead to incorrect simulation results since changes in those inputs won't trigger the block to execute, causing incorrect behavior in combinational circuits.
7. Explain the following statements in Verilog:
    a. casex
       Treats x and z as don’t-care conditions in case expressions.
    b. casez
       Treats only z as a don’t-care condition but x is not ignored.
8. What are Verilog parallel case and full case statements?
    - **Parallel Case:** All cases in a case statement should be mutually exclusive. Failing to make them mutually exclusive can lead to inefficient logic.
    - **Full Case:** Ensures all possible values of the control signal are covered. Missing cases can lead to latch inference.
9. Design 4-bit ripple carry adder in verilog.

```
module ripple_carry_adder_4bit(a, b, cin, sum, cout);
  input [3:0] a, b;
  input cin;
  output [3:0] sum;
  output cout;
  wire c1, c2, c3;
```

```
  full_adder fa0 (a[0], b[0], cin, sum[0], c1);
  full_adder fa1 (a[1], b[1], c1, sum[1], c2);
  full_adder fa2 (a[2], b[2], c2, sum[2], c3);
  full_adder fa3 (a[3], b[3], c3, sum[3], cout);
endmodule

module full_adder(input a, input b, input cin, output sum, output cout);
  assign sum = a ^ b ^ cin;
  assign cout = (a & b) | (cin & (a ^ b));
endmodule
```