1. What are the considerations to be taken when choosing between flip-flops vs latches in a design?
   Flip-Flop
   - Timing: Edge-triggered, which makes timing predictable and simplifies analysis.
   - Data Storage: Ideal for synchronous designs; ensure reliable data storage.
   - Design Complexity: Easier to manage in terms of timing and control.

   Latches
   - Timing: Level-sensitive, which can change state any time the control signal is active, increasing the risk of glitches.
   - Data Storage: Better for temporary storage but prone to race conditions and unintended behaviour.
   - Design Complexity: Requires careful control and can complicate designs, especially state machines.

   Which one is better asynchronous or synchronous reset for the storage elements?
   Synchronous Reset: Preferred for its predictability and integration with the clock signal, making timing analysis easier.
   Asynchronous Reset: Useful for immediate resets but can complicate timing and lead to metastability issues.
2. What logic gets synthesised when I use an integer instead of a reg variable as a storage element? Is the use of an integer recommended?
   Synthesis Logic: Using an integer instead of a reg will not synthesise as hardware storage. Integers are primarily for calculations.
   Recommendation: Always use reg or wire for storage elements to ensure proper synthesis into hardware.
3. Write the Verilog code for Single Port RAM.

```
module single_port_ram (
   input wire clk,
   input wire [3:0] addr,
   input wire [7:0] data_in,
   input wire we,  // Write enable
   output reg [7:0] data_out
);
   reg [7:0] ram [0:15];  // 16 x 8-bit RAM

   always @(posedge clk) begin
      if (we)
         ram[addr] <= data_in; // Write operation
      else
         data_out <= ram[addr]; // Read operation
   end
endmodule
```

4. How do I choose between a case statement and a multi-way if-else statement?
   Case Statement: Preferable when dealing with multiple discrete values for a single variable. It's cleaner and often more efficient for hardware synthesis.
   Multi-Way If-Else: Useful when the conditions involve ranges or more complex logical expressions.
5. How do I avoid a priority encoder in an if-else tree?
   To avoid a priority encoder, ensure that conditions are mutually exclusive and checked in the appropriate order. This can be achieved by structuring the if-else statements to clearly define non-overlapping conditions.
6. What are the differences between if-else and the (“?:”) conditional operator?

If-else

- The synthesis tool generates a more complex combinational logic network, especially if there are multiple conditions.
- Each branch can lead to distinct logic paths, which can affect timing and area depending on how many branches and conditions there are.

Conditional operator

- Synthesized into a multiplexer (MUX) structure.
- A simple two-way conditional, often results in more compact and efficient logic, generally involving fewer gates than an equivalent if-else structure.

7. What is the importance of a default clause in a case statement?
   The default clause provides a fallback behaviour, ensuring a defined operation if no cases match, thus preventing undefined states.
8. What is the difference in implementation with sequential and combinational processes, when the final else clause in a multiway if-else construct is missing?
   Omitting the final else clause can lead to unpredictable behavior or unintended latches if none of the conditions are met.
9. Explain `ifdef `elseif in Verilog with an example.
   ifdef: Allows conditional compilation based on defined macros, including or excluding code.
   elseif: Used in multi-condition checks within if statements.
   `define USE_FEATURE

```
module example;
        initial begin
                `ifdef USE_FEATURE
                        $display("Feature is enabled.");
                 `else
                        $display("Feature is disabled.");
                 `endif
         end
endmodule
```