1. Explain the difference between case, casex and casez.
   case: Standard case statement in Verilog. It matches the exact values of the expression.
   casex: Ignores x and z values in the case expression and case item, treating them as "don't care."
   casez: Treats only z (high-impedance) as "don't care," while x is considered.
2. Explain sequential and parallel blocks with an example.
   Sequential block: Statements inside execute sequentially.
   Example:
   ```
   initial begin
     a = 1;
     b = a + 1;  // b is updated after a
   end
   ```
   Parallel blocks: Statements inside execute in parallel.
   Example:
   ```
   initial fork
     a = 1;
     b = 2;
   join  // a and b are updated simultaneously
   ```
3. Explain the following event-based timing control mechanisms:
   a. Regular event control
      The block is triggered when the specified signal changes.
      Example: @(posedge clk) triggers on the rising edge of clk.
   b. Named event control
      Uses custom named events to control execution.
      Example: event my_event;
      @my_event // Block executes when `my_event` is triggered
4. Explain the conditional statement if and else with an example.
   Used to execute different blocks of code based on conditions.
   Example:
   ```
   if (a == 1)
     y = 1;
   else
     y = 0;
   ```
5. Explain the following looping statements:
   a. while
      Executes the block as long as the condition is true.
      Example:
      ```
      while (i < 10) begin
        i = i + 1;
      end
      ```
   b. for
      Executes the block a specific number of times, controlled by an index.
      ```
      for (i = 0; i < 10; i = i + 1) begin
              sum = sum + i;
      end
      ```
6. How can we disable the naming of blocks?
   To disable a block by name, use the disable statement.
   Example:
   ```
   initial begin : block_name
   ```

```
  disable block_name;  // Disables this block
end
```

7. What is the output?

```
initial begin
        m = 1'b0;
        end
initial begin
        #5 a = 1'b1;
        #25 b = 1'b0;
        end
initial begin
        #10 x = 1'b0;
        #25 y = 1'b1;
        End
initial
        #50 $finish:
```

- At time 0: m = 0
- At time 5: a = 1
- At time 10: x = 0
- At time 30: b = 0 and y = 1
- At time 50: Simulation ends ($finish).