1. How can the infinite loop get created in the looping constructs like forever, while and for.
    - **forever**: This construct creates an infinite loop that will run indefinitely unless there's a mechanism to break out (like a simulation stop condition).
    - **while**: If the condition is always true, such as while (1) or while (true), this loop will never terminate.
    - **for**: An infinite loop can occur if the loop variable is not updated correctly. For example
      for (int i = 0; i >= 0; i++) begin
        // Loop body
      end
2. What are the side-effects of specifying a function without a range.
   If you specify a function without a range:
    - The function can take any integer input, leading to unexpected behavior if the input exceeds the intended range.
    - If bounds checking is not implemented inside the function, it might lead to simulation errors or incorrect results.
3. What happens when we use a tri-state logic in a chip design?
    - Tri-state logic allows multiple drivers for a signal but only one driver can be active at any time.
    - When a tri-state output is disabled (high-impedance state), it prevents signal contention, allowing other devices to drive the bus.
    - However, incorrect configurations can lead to contention if two drivers are enabled simultaneously, causing potential damage to the components and unpredictable behaviour.
4. What happens if you don't have a final else clause in an if-else statement?
    - The synthesis tool may infer a latch for the output variable, which means it will hold its previous value if none of the conditions are met, potentially leading to unintended behaviour.
    - This can also introduce timing issues and may complicate debugging.
5. What happens if you miss a default clause in a case construct?
    - If none of the case values matches, the output will remain unchanged, which can lead to unintended behaviour (e.g., holding previous states or values).
    - This may also result in latch inference if the output variable is not driven by any other logic in all possible cases.
6. How can unintentional deadlocked situations occur during simulation?
    - Cyclic dependencies where two or more processes are waiting for each other to release resources.
    - Lack of proper event triggers to allow processes to proceed, leading to them waiting indefinitely.
7. Having a programmed loop that does not move simulation time. How can you avoid this situation?
   To avoid this situation:

- Ensure that the loop contains time-related operations, like delays (#), or events that trigger changes in simulation time.
- Use constructs like @(posedge clk) to introduce clock cycles that will move the simulation forward.

8. What happens if you leave an input port unconnected that influences a logic to an output port?
    - The unconnected input will typically assume a default value (logic high 1 or low 0, depending on the simulation tool).
    - If the input is used in logic, this could lead to unexpected behaviour, such as always being in one state due to the assumption of the default value.
9. What if you don't connect all ports during instantiation?
    - Unconnected ports will default to unknown values (x), potentially leading to incorrect logic operation.
    - This can cause simulation failures or unexpected results during synthesis.

10. What if you forget to increase the width of state registers as more states get added to a state machine?
    - The state machine may not be able to represent all intended states, leading to incorrect state transitions or unreachable states.
    - This could result in undefined behaviour or synthesis errors if the number of states exceeds the capacity of the register.
11. What if there is an implicit 1-bit wire declaration of a multi-bit port during instantiation?
    - You will face truncation issues. Only the least significant bit will be connected, and the remaining bits will be lost.
    - This will likely lead to erroneous outputs and difficult-to-diagnose bugs.
12. What happens when the same variable is used in two loops running simultaneously?
    - Race conditions may occur, leading to unpredictable behaviour.
    - One loop may overwrite the variable's value before the other loop can use it, causing erroneous results.
13. What if multiple processes write to the same variable?
    - It can lead to contention and unpredictable results. Only one driver can drive a wire at any time, and if multiple drivers exist, the behaviour will be undefined.
    - It’s essential to ensure that the variable has a unique driver or manage access appropriately using enable signals.
14. What are the side effects of specifying delays in assignments?
    - Simulation mismatches where the timing of signals does not reflect the actual hardware behaviour.
    - Difficult debugging due to non-deterministic behaviour introduced by delays, especially if not carefully controlled.